Archive

The Subscription Magazine for *Archimedes* Users

February '88

Vol. 1 No. 5

Price £1.20

BASIC

OS Interface

The PC Emulator

Automatic Re-Configuration

File Search Utility

Arthur 1.2 is Here!

Graphics Galore!

Reviews & Routines

4 Extra Pages this Month!!

Emphasis on Graphics

The main emphasis in this month's issue is graphics. There are reviews of art and drawing packages, some suggestions of how to use the mouse in your own drawing/art applications, a design-your-own-pointer program and some routines for drawing those smart looking 3-D "buttons" you get on Clares software.

Value for money?

A recent review on Micronet compared Archive with RISC User and came to the conclusion that Archive was better value for money. If you compare 10 issues at 31 pages for £12.50 with 12 issues at 49 pages for £10, it's difficult to refute it. (That's 4.03 pence/page as against 1.70 pence/page.) Even if you add £8 for the cost of subscribing to our Technical Help Service (RISC User's telephone help service is free) you still get only 3.06 pence/page!

But what is really important is not the fact that we **do** offer better value for money but **why** we do so. The reason is plain to see – the majority of the material which RISC User publishes is written by Beebug's own staff whereas virtually all the material in Archive is written by **YOU!** The moral is that if you want Archive to continue to be the best value for money, you need to keep the articles coming.

(You can see by the way I've had to squeeze the articles into every available space this month that there is a steady supply. I hope it doesn't look too cramped.)

Quality as well as Quantity

It's OK to be the best value for money in terms of quantity, but are we the best in terms of quality? Well, it depends to some extent on the work that I do as editor, but it depends chiefly on the quality of the contributions we get from you. Don't let that put you off by thinking that your work won't be good enough – let me be the judge of that – that's what editors are supposed to do, I think! The only thing I do ask is that you will understand if you submit something and I don't actually publish it. It may not be because we have too many articles dealing with a particular area of interest.

But don't let me put you off. Even if you haven't written for magazines before, why not have a go? I used to think how terribly exalted all these "Experts" were who wrote in magazines – then I became one and my illusions were shattered! If you've got something interesting to tell us about, let me know. If you think your standard of written English isn't good enough, don't worry; it's my job to act as a quality control on that side of things too.

1,000 up!

We now have over 1,000 subscribers – nothing compared with RISC User's subscription list, but if you're enjoying the magazine and finding it helpful, and Sue and I are paying our bills, we're all happy! If you're NOT finding Archive helpful and/or interesting, please tell us why. If you haven't the time to write a letter, a quick phone call would do. I do welcome criticism (as long as it is constructive) and accept it very graciously even if I don't agree – so you needn't fear that you will get your head bitten off!

Once again, a very big thank you from Sue and myself,

Paul Beverley

Archive

Volume 1 • Nº 5 • February 1988

Contents

News, News, News......

• Hard Disc + Podule – Available February – "probably late in the month". A440s are coming out of the factory in dribs and drabs, but there's still a huge backlog of unfulfilled orders.

• PC Podule – the word from David Bell, Acorn's Hardware Product Manager is that the PC Podule is "still on hold" and he should know!

• RS423 Saga – Part 6… The latest unofficial news from Acorn is that they think they've worked out how to fix the problem. Previous fixes have been specific to certain RS423 chips and don't work with other chips. (They've used chips from four different sources so far.) They are now working on a software patch that should work for all machines. Let's hope so – I can't use mine at all at the moment.

• "All the remaining 1.2 operating systems were sent out last Friday" (22nd January). Thus said Acorn Technical Support to one of our readers. Judging by phone calls from other readers, I gather that all 0.2 owners have had their 1.2's, but not all 0.3 owners. If you still haven't had yours, arm yourself (oops, sorry) with your serial number (which is **underneath** the computer, so you have to take the monitor off first!) and ring Acorn.

Hardware & Software Available

Norwich Computer Services is now an Acorn dealer, so we will be stocking some of the smaller items of Acorn hardware and software. We can offer 7.5% discount to members as with other things we sell, but please ring to confirm if you are not sure whether we stock a given item.

• CC ROM Podule £53 (Full price: £56.35) or £63 (£68.94) with battery back-up. (**Should** be available by the time you get this magazine!)

• Back Plane (£44.85 – £42 now available through Archive)

• I/O Podule (£90.85 – £84 now available through Archive)

• IFEL Electronics have produced their own backplane for two podules – £35.65 inc VAT.

• BBC to Archimedes Conversion Utilities. If you've got some precious BBC programs that you are trying to get working on the Archimedes, help is at hand. RESOURCE are marketing a whole suite of programs (written by Acorn software engineers) to help you get your BBC programs working properly under the 6502 emulator. Price: £15.95.

• Teletext and TV Adapter. Solidisk are launching a programmable TV tuner with Teletext adapter for the Archimedes which is on special offer until the end of March at £79.95 inc VAT and carriage. (Normal price £103.) They've sent us one to have a look at, so we'll let you know what we think next month.

• Brainsoft have now released a new improved version of their Archimedes-BBC link package. The new package will access ADFS on the BBC end as well as DFS and the Archimedes desktop has a 'Big Ben' clock and a 40-function scientific calculator. Then as an optional extra, for just £5, you can have a text editor for use with compilers. It will allow you to edit 2,000 lines of text, and have two windows open on the text at the same time. The basic software is only £8, £14 with the RS423 lead, but add £5 to either price for the editor. We now have a phone number – see Factfile at the back of the magazine.

• Beebug have also released a BBC to Archimedes link which also works on both DFS and ADFS but which has a 3 metre cable instead of a 1 metre cable and "more sophisticated software" according to a comparative review in RISC User magazine.

Help!!!......

• I have a Viglen 20 Mbyte hard disc. How can I get it linked up to the Archimedes? Acorn say that they will not sell the hard disc controller podule separate from the drive unit. (Several people have asked the same thing.)

Actually, one or two folk are trying to work out if this is possible. If anyone has any information, please let us have it so that we can pass it on – it could save a number of folk, who already have hard discs, a lot of money!

• Does anyone know anything about the FP Emulator? Can it be used from BASIC? C? Fortran etc? or only from ARM code programs. Yes, I know there is information about it in the Programmers' Reference Manual, but I didn't understand a word of it. Can someone translate it for the relative layman?

• Anybody got any information about the RAM filing system? Is there one?

• Has anyone got a screen dump for the Apple Laser-Writer?

• "The MS-DOS emulator is not all that easy to use", says one reader. Can anyone give us a bit of help on this one, please? I have no experience with MS-DOS at all and haven't bothered to buy the emulator (especially after they put the price up!) so I can't offer any help myself.

• Matthew Treagus is interested in hearing from anybody in education regarding the Archimedes, with reference to queries about compatibility and usefulness or any comments/ problems. Contact him on YMT071 – that's the Hardley School TTNS Mailbox. He will shortly be running an Archimedes database there. Your comments and ideas are most welcome. Prestel mailbox number 703893596. Mail: 30 Hampton Lane, Blackfield, Southampton, SO4 1ZA.

• One reader is having problems with getting VDU23,26...etc <fontname> working. It either gives the error message "bad filename" if no name is given or "Unknown or missing variable" if a correct name is given. Any clues?

• Alex Hopkins of West Ruislip asks if anyone is writing a "multi-screen converter" program! He wants to be able to convert screens between modes 1, 9 and 13 (and 19 & 20) and between modes 0, 8, 12 and 15 (presumably upwards conversion to mode with larger numbers of colours). The idea being to convert the screen pixel and colour perfectly to allow enhancements to be made by adding extra colours.

• Is there a User Group is S Wales? Anyone interested in forming one? If so, contact Roy McNamara, 19 Priory Oak, Brackla, Bridgend, Mid Glamorgan, CF31 2HY.

Help Offered

• I misunderstood Dr Fielding's offer of help last month. He has translated some matrix algebra routines **from** Fortran 77 to BASIC V. Contact him on 061-228-6171 Ext 2410 for more details.

• More help on the RS423. Brian Carroll of Cottage Micro Services can offer a description of a cable for linking various micros on the serial port using a "null modem cable". The cable evolved by trial and error and allows the following six interconnections with hardware flow control: Either (Amstrad PCW 8256 with CPS8256 port) or (BBC Micro B, B+ or M) can be connected to either (Wren Executive System) or (Archimedes) or (BBC Micro B, B+ or M). For details, send an S.A.E. to Brian Carroll, 42 Manor Road, Aldershot, Hants, GU11 3DG.

• Andrew Featherstone is offering a text editor, written in BASIC, to any reader who wants to send him a blank, formatted disc and an S.A.E. However, I think if you get it from him and find it's useful, he'd not be averse to receiving a small donation! His address is 31, Mallard Close, Twyford, Reading, RG10 0BE.

Matters Arising...

• **Graphics Demonstration Disc.** Contrary to what I said in the instructions, you **can** edit the names on the list of programs without having to delete and re-enter them. What you do is press <space> to remove the protection, press <return> and you can then type in the new program title over the old one and press <return>. If it's not the title you want to change, just press <return>. Then type the new filename over the old one and press <return> to finish. Finally, press <space> to re-protect the list.

• Some of you were having problems with the **RS423 file transfer program** from Archive issue 1. The old BBC micro didn't like *FX156,&56,0. The reason is that in earlier operating systems, all parameters for FX calls had to be in decimal, so you should use instead, *FX156,86,0.

• Those of you who typed in the program **"Archive Colour Blender"** in Issue 3 (or bought the program disc for that issue) may well agree that one minor drawback, in an otherwise excellent program, is the slow speed of writing the 40 column text! The author, Ian Nicholls, has taken an improved BASIC routine for PROCosword, sent in by Gavin Halliday, and produced a much faster ARM assembly language routine based on it. This speeds up the 40 column text printing quite considerably.

We have produced a listing of the changes which, provided you stick to the line numbers given, will automatically insert new lines and overwrite old lines which need changing.

There are two ways in which you can make the changes: one is to load the original program "BLENDER" and then just type in the program lines below. The second way is to clear out any existing program with NEW, and type the lines in as if they were a program. Next SAVE this program with the name "MODS_BASIC". Now type the command *SPOOL MODS <return>, followed by the command LIST <return> and, lastly, the command *SPOOL <return>. You have now created a file called "MODS" which contains the new lines in ASCII format. The final step is to load the original program "BLENDER" and type the command *EXEC MODS <return>. This will then merge the changes into "BLENDER", also giving it the new title "BLENDER_2".

With the second method, it is easier to correct any mistakes in your typing, because you only need to look through the new lines SAVEd in the program "MODS_BASIC".

(The other alternative is to buy the program disc for this issue – a lot simpler!)

```
  10 REM > BLENDER_2
  30 REM Ian Nicholls, 8/1/88
  75 DIM code 400,buffer 30
  80 DIM flag%(6),A$(6),x%(6),
                            y%(6)
  92 PROCassemble
  94 FORI%=1TO15
  96 READ bits?I%
  98 NEXT
 110 FORI%=1TO13:READc%,r%,g%,
      b%:COLOURc%,r%,g%,b%:NEXT
1320 DEFPROCprint_40(F%,G%,
                          B$,p%)
1330 H%=buffer:$H%=B$
1340 IFp%=0 THEN
1350 CALL double_width
1360 ELSE
1370 L%=F%:M%=G%
1380 FORJ%=0TO2
1390 COLOUR(5+J%):F%=L%
                     :G%=M%+5*J%
1400 CALL double_width
1410 NEXT
```

```
1420 ENDIF
1450 DEFPROCassemble
1455 WriteC=0:WriteI=256
1460 FOR pass=0TO2 STEP2
1465 P%=code:[OPT pass
1469 .double_width
1470 STMFD (13)!,{14}
1471 SWI WriteI+31
1472 MOV R0,R5
1473 SWI WriteC
1474 MOV R0,R6
1475 SWI WriteC
1476 ADR R2,bytes
1477 ADR R4,bits
1478 .print
1479 ADR R1,char
1480 LDRB R0,[R7],#1
1485 CMP R0,#13
1490 LDMEQFD (13)!,{PC}
1495 STRB R0,[R1]
1500 MOV R0,#10
1505 SWI "OS_Word"
1510 SWI WriteI+23
1515 SWI WriteI+130
1520 MOV R1,#0
1525 .loop1:LDRB R3,[R2,R1]
1530 MOV R3,R3,LSR#4
1535 LDR R0,[R4,R3]
1540 SWI WriteC
1545 ADD R1,R1,#1
1550 CMP R1,#8
1555 BNE loop1
1560 SWI WriteI+23
1565 SWI WriteI+131
1570 MOV R1,#0
1575 .loop2:LDRB R3,[R2,R1]
1580 AND R3,R3,#15
1585 LDR R0,[R4,R3]
1590 SWI WriteC
1595 ADD R1,R1,#1
1600 CMP R1,#8
1605 BNE loop2
1606 MOV R0,#130
1607 SWI WriteC
1608 MOV R0,#131
1609 SWI WriteC
1610 B print
1615 .char:EQUB 0
1620 .bytes:EQUD 0:EQUD 0
1625 .bits:EQUD 0:EQUD 0:EQUD 0
                            :EQUD 0
1630 ]
1635 NEXT
1640 ENDPROC
1765 DATA3,12,15,48,51,60,63,192
     ,195,204,207,240,243,252,255
```

Monitor Information

Thanks for all the response to our request for information about monitors. Here is a summary of the responses so far. These are the individual readers' views, not Archive's.

• Novex Amber screen 12/800 "produces good readable text in both 80 and 132 character modes".

• Rediffusion System Alpha (analogue colour monitor) will work with the Archimedes but not really high enough resolution. 132 column modes are almost unreadable although it does give the full 256 colours.

• Grundig Super Elite 14" TV/monitor (as converted my Newark Video Centre) works well and the display is very crisp. The reader who told us this, was keen that we should publish the information because he was wrongly advised by several dealers, including Beebug who tested it and "proved" that it didn't work. All you need is the correct SCART lead, available from Watford Electronics, and away you go! (By the way, what **does** SCART stand for?)

• NEC multisync monitor gives an excellent display and a modified Microvitec is also good but it cannot, of course, display the three very high resolution modes. A

Hints and Tips...

• **Text files for BASIC.** I always used to write my BASIC programs in Wordwise and put an AUTO command as the top line then go into BASIC and NEW and then *EXEC <filename>. But I've just discovered (what I'm **sure** you all knew already) that BASIC can do it for me. You don't have to put AUTO at the top you just say *BASIC -LOAD <filename> and it will load the text version of the program and put line numbers onto it. However, as with the *EXEC technique, you cannot use long multi-statement lines or you will get the "Line too long" error.

(One reader, who was using Arthur 0.2, found that text files longer than 32k got corrupted, but I tried it on my system with BASIC 1.02 and Arthur 1.2 and it seemed OK.)

• **Fast screen save.** This is obviously something which a number of you have been thinking about. I got a lot of response to my pleas for help. Several of you gave ways of *SAVEing the screen information from the appropriate bit of RAM and re-loading it there, though this obviously will not bring with it the palette information or the correct mode. For example:

```
DEF PROCscreensave(filename$)
LOCAL K%
CASE MODE OF
  WHEN 0,1,4,5,6 : K%=20
  WHEN 2,3,8,9,11,18 : K%=40
  WHEN 7,10,12,13,14,19 : K%=80
  WHEN 16,17 : K%=132
  WHEN 15,20 : K%=160
ENDCASE
!&80=149:!&84=-1
SYS 49,&80,&88 : REM
OS_ReadVduVariables
OSCLI"SAVE "+filename$+" "+STR$
    ~(!&88)+" +"+STR$~(K%*1024)
ENDPROC
```

This takes only a second or two to save the screen. It can be reloaded by typing *LOAD filename. The program saves the whole screen, not just the graphics window. You must not have scrolled the screen prior to saving or re-loading, but a CLS before re-loading will cancel the scrolling. Also, you must be in the right mode for the screen you are about to load and you need to have the same screen colour definitions set up.

What we have been promised though is a module, which we hope to publish next month, which will give you legal *commands (or SWIs) for loading and saving screen information **including** palette and mode and you will find that it is several times faster than the existing SCREENSAVE routines.

• **Extra help on BASIC...** (OS1.2) If you type *BASIC -help<return> you get a bit more help information about BASIC. (But does anyone know what the last two bits refer to? i.e. what is an "in-core" program? Does it mean you can have more than one program in memory?)

• **ADFS:** If you name all your discs with *NAMEDISC, the Archimedes will remember the previously mounted discs. So, for example, if you put in a disc called TESTING and you *MOUNT it and then put in a disc called BASICPROGS and *MOUNT that, you can then see the directory of TESTING without putting the disc back in again. All you do is type *MOUNT TESTING and you can *CAT it. If you then want to look at the disc that is in the drive, you will of course have to *MOUNT again. The only problem occurs when you have backup discs because they will have the same name – this gives "Ambiguous disc name".

• **Sprite Editor.** "To create a new sprite..." It took me ages to find out how to create a new sprite. The manual tells you to click on the "Create" box and "you are then prompted for a name and a mode". I was expecting a prompt something like: "Please enter name:". I saw the words "Create sprite" appear at the bottom of the screen and assumed it wanted some sort of confirmation that that was what I wanted to do, but whatever I did, the words disappeared again! In fact **that** is the actual prompt that the manual was referring to! All you do is type in the new name and press <return>. You would have thought they could have put in a more obvious prompt than that – or maybe just switched on the cursor to indicate that it was time to type something rather than click on the mouse buttons, but perhaps I have been spoiled by using the Apple Mac where prompts are rather more obvious and logical! Huhh!

• **Reading the catalogue**. An easier way to read the contents of a catalogue, rather than reading it off the screen (see Gus Gem's article last month) is to use SYS "OS_GBPB" as follows…

```
DIM C% 999
!&80=&2A00
SYS "OS_GBPB",9,&80,C%,999,0,
                 999 TO ,,,ENTRIES
```

This stores the filenames and sub-directories of the current directory at the address C%, in alphabetical order and each terminated by a zero byte. The number of entries is stored in the BASIC variable ENTRIES.

• **Neater than OSCLI…** SYS 6 (equivalent of FX calls) is much neater than using OSCLIs: e.g. Matthew Treagus' screen bank switching would look neater if, instead of using OSCLI ("FX112,"+STR$N%) he had used SYS 6,112,N%. If you don't like the anonymity of SYS 6, you can use SYS "OS_Byte",112,N%. (See Gerald's article on page 12.)

• **Beebug's Masterfile** can be made to work on the Archimedes **without** using the emulator – it just needs a very few modifications – contact Dr Peter Catermole at Winchester College, Kingsgate Road, Winchester, SO23 9PG.

• **BBC ROMs** – Continuing the saga of poking various bytes to get BBC ROMs to work under the emulator, View B3.0 should apparently have A8A1, 2 and 3 poked to &EA and Viewspell 1.0, &820F, 8210 and 8211 likewise.

• When **setting the clock**, the setting box sometimes covers the face of the clock, so point at the bottom right hand corner of the clock box and then press the middle mouse button.

• **Plugging modules in and out.** It's all very well to have lots of modules in the 1.2 ROM but, at power up, several of these modules are installed into ram which cuts down your available memory. Exactly which ones are installed is set by various bytes in the CMOS ram. To find out what all the modules are and whether they are actually active, type *ROMModules. (or just *ROM. for short.) You can "unplug" the modules by using *UNPLUG <modulename> and it stays unplugged until you "insert" it again with *RMREINIT <modulename>. It seems that the extra space does not become available until you press <ctrl-break>. If you type *UNPLUG without a module name it tells you which modules are currently unplugged.

• **How's this for a screen dump?** (sent in by Ken Yeardley) It is for mode 0, it works on the NEC PC8023, it can be put on a function key and it only takes 118 seconds.

```
VDU2,1,27,84,1,&31,1,&36: FOR
A%=&1FDCFB0 TO &1FDCFFF:VDU2,
1,27,83,1,&30,1,&35,1,&31,1,&32:
FOR B%=A% TO A%-20400 STEP-&50:
VDU1,?B%,1,?B%: NEXT: VDU1,10:
NEXT: VDU3
```

The reason it is so short is that the NEC printer has the MSB and LSB of the byte that defines which pins hit the paper, the opposite way round from Epson printers. (See page 9 for a sample screendump.

• **Beware the microwave!** One reader was having problems with data corruption on the RS423 but it turned out to be only when his wife was using the microwave oven four yards away, the other side of a brick wall!

• Here's a little routine sent in by Mike Hobart which uses the debugger module for **disassembly** and **memory dumps**. It doesn't quite rival Toolkit, but it's a bit cheaper!

```
INPUT"Disassembly or Memory
     dump? (D/M)",R$
CASE R$ OF
  WHEN "d","D" : N%=FNi:REPEAT
     PROCx("I",N%," +10"):UNTIL
     FALSE
  WHEN "m","M" : N%=FNi:REPEAT
     PROCx("",N%," +FF"):UNTIL
     FALSE
ENDCASE
END
DEFPROCx(S$,M%,L$)
WHILE INKEY(-99)
  OS_CLI("MEMORY"+S$+" "+
                STR$~(N%)+L$)
 N%=N%+100
ENDWHILE
ENDPROC
DEFFNi
INPUT"From(Address &nnnnnn)"'N$
PRINT"Press <space> to scroll"
IF ASCN$<>ASC"&"THEN N$="&"+N$
=EVAL N$
```

• **Desktop on 1.2.** (All the remaining H & T are based on material sent in by Bruce Roberts.) As an alternative to **double clicking on <select>** to open a file or run an application, try a single click on the adjust button – it has the same effect.

The quick way to **open a diary** (or notepad), rather than opening the diary window first and then pressing <menu>, is to double click on the diary icon on the disk directory – the window opens automatically.

You can now **save diaries and notepads** with whatever name you like and you can also click, or move with the cursor keys, and put text anywhere in the window.

The following keys can now be used on **diary and notepad:**

<insert>	Insert single character
<copy>	Delete character to the right of the cursor
<ctrl-insert>	Insert a line above the cursor
<ctrl-copy>	Delete a line
<shift-left/right>	Move 4 characters sideways
<ctrl-left/right>	Move to the end of the line

Notepads can be created from within View by saving a file (up to 100 lines, 78 characters wide) and then doing a *SETTYPE <filename> &FEE. Existing notepads can be edited with View provided you use READ <filename> instead of LOADing it and WRITE <filename> instead of SAVEing it.

• **After using the desktop,** the function keys do not produce the normal programmable strings or ASCII values. It seems that this is because Acorn have not stuck to their own rules (see Programmers' Reference Manual, page 488) because they do not call SWI Wimp_CloseDown. The "fix" published elsewhere of typing *FX225,1 is fine if you only want to restore the strings to f0 - f9. To get f10 - f12 back, you need *FX221,1 but the simplest thing seems to be to make sure that everything is back to its default setting by using a program, perhaps called QUIT, which you can run from the desktop instead of clicking on the exit icon. To set the default of all the combinations of <ctrl> and <shift> with all the function keys, you need:

```
*FX225,1
*FX226,&80
*FX227,&90
*FX228,0
*FX221,1
```

```
*FX222,&D0
*FX223,&E0
*FX224,&F0
MODE 0
*BASIC
```

If you don't put in a MODE statement, you'll find yourself typing invisibly! The *BASIC means you come out into BASIC but with no program in memory.

As an alternative, you could miss out the call to BASIC and put:

```
W%=GET
*MOUNT
*!BOOT
```

The idea would be that you would click on this program, change to the disc you want to boot up, and press a key. You may want to include a prompt to the user to "replace disc and press space bar" or somesuch. The use of *!BOOT means that it doesn't matter if the boot file should be *EXECed or *RUN – either will be done by the *!BOOT according to the file type.

• **Avoiding the desktop.** If you don't want to go into the desktop every time you switch on or press <ctrl-break>, do a *CONFIGURE LANGUAGE 4 and press <ctrl-break>. When you want to go into the desktop, type *DESKTOP or *DES. for short. If you change your mind, *CON. LANG. 3 will bring you into desktop at switch on or LANG. 0 will start you in the supervisor mode.

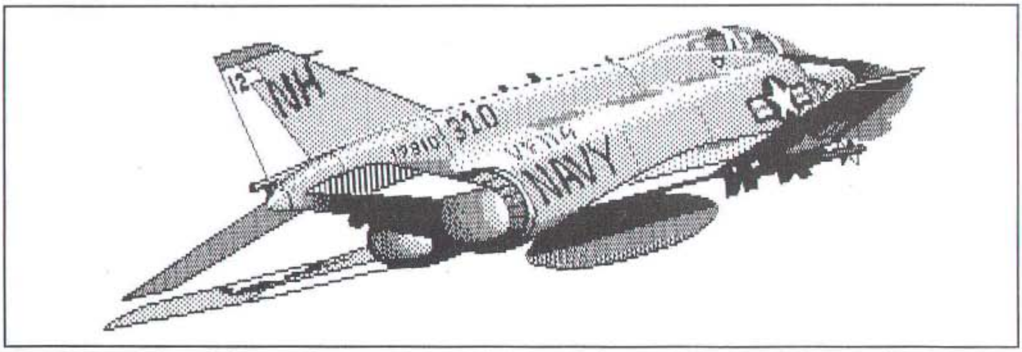
• **View boot file.** Here is a more sophisticated boot file than the one we gave originally:

```
*ECHO<22><3>
*| selects mode 3
*GOS
*65ARTHUR
*GO F800
*LOAD VIEW 8000
*GO 8000
*ECHO<19><0><16><48><48><240>
*| sets background colour
*ECHO<19><1><16><240><240><208>
*| sets foreground colour
*ECHO<19><2><24><240><128><0>
*| sets border/line colour
SET FI
*| sets Format and Insert modes
NEW
```

• **Avoiding *MOUNT.** To avoid typing *MOUNT each time you put a new disc in the drive, you can set an alias with, for example, *SET ALIAS$? MOUNT|M CAT so that when you type *?<return> it does the mount and the catalogue all in one.

• **Centronics GLP screendump.** The *HardCopyMX **nearly** works because the printer is Epson-compatible-ish! All you need to do is issue, from BASIC, the following codes:

VDU2,1,27,1,65,1,8,1,27,1,50,3



King Arthur is Crowned or 1.2 At Last!!!!

Matthew Treagus, Adrian Look et al

Matthew starts… Finally it's here, well mine is anyway! Arthur 1.2 has surfaced in Acorn's sea of excuses. On the 11th December I was told "Arthur 1.2 will be posted this week.. but allow for delays due to the seasonal postage problems" Hmm! even our good old GPO are not **that** slow. Anyway my envelope was post-marked the 7th January 1988. I eagerly tore open the envelope and found an unexpectedly large box which contained the four 1.2 Arthur Chips and Welcome Disc, fitting instructions, upgrade news sheet, Issue 3 User Guide (with index and corrections), Issue 3 Welcome Guide describing the new music editor and padded envelope and postage paid label. Paused for confused thought. Acorn have asked all 0.2 Arthur owners to return the chips for some reason. *(I guess it's because 27512's cost at least 5 pounds each! Ed.)*

But enough of this dribble what have you got and can you fit it easily? Well it is easy to fit and you've got a lot. All the following items are now in ROM: Utility Module, File Switch, Desktop, BASIC, Debugger, System Devices, ADFS, Econet, NetFS, NetPrint, ARMBasicEditor, Window Manager, FontManager, SpriteUtils, SoundDMA, SoundChannels, SoundScheduler, WaveSynth, StringLib, Percussion, Hardcopy, International Module and the Podule Controller.

The BASIC Editor has been improved in that the flashing cursor can now be seen, two windows can be opened on one program, a help key is instantly to hand and program info is also available. The *WIPE and *COPY commands have been changed and are now a little less violent. Each time you specify a wipe or copy it will ask for confimation of each file before it acts on it. To avoid this you can add the suffix ~C. This means that it does **not** ask for confirmation.

A *COUNT <file spec> command has been introduced which tells you the number of bytes occupied by the specified directories and/or files. *COUNT * gives the sizes of all the directories within the current directory and *COUNT * V gives the sizes of all the files **and** directories within the current directory **and** within all the sub-directories. Working screen dumps are resident (for Epson & compatibles, anyway), the ADFS works perfectly as far as I know and all the functions a normal(?) programmer could ever want are resident in ROM. The Desktop and accessories are instantly to hand and even the RS423 works perfectly. *(Are you sure? Mine doesn't! Ed.)*

Can it really be true that Acorn have finally cracked the nut of high powered computers or is this some futuristic dream from which I will awake to find my Commodore VIC20 still perched on my desk under a flickering old telly. But let's find those bugs in Arthur 1.2 there must be some in it…

Speed Changes

Adrian & Ed have also been doing a bit of playing around with the new 1.2 operating system and in particular they have notice something of an improvement in speed of plotting and drawing (at least when compared to the 0.2 OS.)

Having noticed how much faster some of the programs on the Shareware Graphics disc seemed to work, we wrote various test routines for the plotting and drawing and tried them out on 0.2 and 1.2. Here are the results:

Function	Speed improvement ($TIME_{0.2}/TIME_{1.2}$)
PLOT69	1.4
LINE	1.6

RECTANGLE	6.9	(sic)
RECTANGLEFILL	2.1	
ELLIPSE	1.3	
ELLIPSE FILL	1.4	
PLOT SPRITE	1.7	

We too thought that 6.9 couldn't be correct but we didn't want to take the chips in and out too often. Still, for the sake of accuracy we tried it again with a slightly different program and still got roughly the same ratio of 7:1. So then we tried it by DRAWing rectangles: 6.3. Why then should LINE be about the same as the others and repeated DRAW commands different? We tried MOVE and DRAW in place of the LINE command and that gave a ratio of 1.6:1. Most confusing!

All we can really say is that the new operating system is at least 30% faster and, in certain circumstances, up to 7 times as fast! Not a very scientifically satisfying conclusion; never mind.

A Calculator with Keys!

The other thing we noticed that pleased me no end, is that the calculator will now accept input from the keyboard. Having been used to being able to hammering out calculations on the Mac desktop, I thought that only being able to use the mouse made the calculator into little more than a clever toy.

Bug or Feature?

Here we begin a list, hopefully not too long, of the problems with Arthur 1.2 / BASIC 1.02.

• One 'feature' of the Podule Manager (1.03) is that when you access one of the podules, the clock stops! So don't bother trying to use the internal timer to time the access to the ROM filing system. If you do, you will find that both the Acorn ROM FS and the Computer Concept ROM FS take 10 millisecond to load a 64k file!

• The command "*HELP FileCommands" omits reference to *CDIR, *ACCESS, *INFO, *RENAME, *COPY, *EX, *OPT and *IAM whereas using "*HELP Commands" gives all of the filing system commands.

• There seems to be some problem with mode 20 (not that I have any way of testing it!). The mouse pointer splits up when it gets to the right hand edge of the screen, and the HardcopyFX module misses off the right hand edge of the picture.

Do tell us all your joys and sorrows with 1.2 – they say it helps to talk about these things! A

Programmers' Reference Manual Additional Changes

Here are some changes **in addition to** those already given in the Reference Manual Addendum.

• Page 78	For p=25, it should read logical colour of **pointer**, not cursor.
• Page 89	In the bit map, "left pixel" and "right pixel" should be the opposite way round.
• Page 191	In the table, "bits" should be under "Stop" not "Parity"
• Index	"OS_Byte – index of" should be page 644, not 643
	"OS_Word – index of" should be page 647, not 646. A

The BASIC / Operating System Interface

by Gerald Fitton

This article was written partly in response to the plea of non-BBC users to explain about FX's and OS_Words which are rather new and strange to them. Gerald is, as you will be able to tell, a BBC user but whatever your background, you should hopefully be able to understand and appreciate your Archimedes better after reading this article.

In order to understand how BASIC and the Operating System inter-relate on the Archimedes we need first to understand some basic concepts of the structure of computers.

Languages and Environments

To start off with, we need to get the idea that programs are written in 'Languages' and executed from within an 'Environment'. Confusion sometimes arises because the word BASIC is used both for the language and for the environment in which the program is run. A couple of examples may help to clarify the difference.

1. A BASIC program can be written, i.e. prepared, using a text editor environment such as ARMBE or Wordwise but it can only be executed from within a BASIC environment.

2. There is nothing to stop you writing a program containing commands peculiar to the BASIC V language on a Master, BBC model A or B or (even) an Electron or IBM. However, it will not **run** on the model B because the earlier BBC Micro's BASIC environment will not support the new commands. However, you can transfer it to an Archimedes and then run it there.

Environmental Shells

Most machine manufacturers supply software with the machine which creates a usable working environment called the Operating System (OS). The software developer writes a program using (documented!) OS commands to create an outer 'user friendly' environmental shell such as BASIC or WIMPS (Windows, Icons, Pointers and Mouse). The least friendly environment is inside the Central Processing Unit (CPU) and written in a language called microcode which is understood only by the CPU and its creator. Microcode programs consist of instructions which manipulate switches (logic gates) within the CPU. These CPU instructions are accessed indirectly by the assembly language programmer who writes mnemonics using a text editor environment. The assembly language program is then compiled into machine code language instructions (these are just 8, 16 or 32 bit numbers representing logical voltages applied to the pins of the CPU) which are interpreted into microcode and executed within the CPU. Languages such as BASIC are interpreted into machine code instructions as the program is run. Other languages, such as 'C' and PASCAL, are written in a text editor environment and then compiled into a program which is saved as machine code instructions. This machine code program can then be *RUN from within the OS environment.

Upward and Downward Compatibility

When the CPU is upgraded, (e.g. a 3 micron RISC chip is replaced by the 2 micron version) the microcode of the chip will change, new machine code instructions will be added but the old machine code and their corresponding assembly language instructions will not change. The expression 'upgrade compatible' implies that new instructions have been added but the old ones have not been deleted. This is one of the advantages of designing software in environmental shells – programs written for one machine can be run on another upgrade compatible machine. Higher level languages, such as the graphics language GEM Paint, BASIC, or WIMPS can look the same to the end user on widely machines. This is achieved by designing an interface between the high level environment and the inner environment which ensures that, independent of the hardware, the same input produces the same output.

Perhaps the two most important environments for the average Archimedes user are BASIC V which is used as an outer shell environment and the inner environment called the 'Operating System'. For the software developer, it is important that the programs they write for one version of the machine will run on later versions that might not yet have been designed. The school or college with many machines also depends upon the manufacturers not to make their old software redundant when the machine is upgraded. The commercial user is often not so worried about the price of new software but is very concerned that any data files created with the old system can be transferred to the new.

BASIC Programs

The word BASIC is an acronym for Beginners All Purpose Symbolic Instruction Code. Although there are many dialects of BASIC, the original was never meant to be used on a 'real' system and, since it was written before the use of graphics became widespread, it is text oriented. As a consequence, each machine designer has introduced his own graphics commands. Acorn use DRAW and PLOT as well as an extensive set of VDU commands which are not in the original (generic) BASIC.

One of the problems with the original BASIC is that it depended heavily on the use of line numbers. This encourages 'unstructured' programs. Unstructured programs are difficult to debug, read, understand and develop. GOTO <a line number> is all to easily mis-used as a way of causing a change in a sequence of instructions. Acorn have added multi-line functions and procedures to their dialect of BASIC. These functions and procedures can be called by name (or by a 'label', as we say). In the Archimedes (BASIC V) Acorn have also introduced the LIBRARY and INSTALL commands which further enhance what has been described as the most advanced dialect of BASIC in the world. Perhaps one of the greatest strengths of the Acorn machines is the excellent interface between BASIC and the OS. Almost any OS routine can be executed from within the BASIC environment.

'*' Commands

All OS '*' commands can be called from either the BASIC or OS environment. To leave BASIC, type QUIT<return>. You will then be in the OS environment (or Supervisor mode) having left behind such commands as PRINT, REPEAT and END. From the OS you can return to BASIC with *BASIC. From within BASIC V there are three ways of calling an operating system routine – putting them as * commands, or using OSCLI or SYS. The simplest is through a *-command such as *HELP which 'works' both in BASIC and in the OS environment. Try *HELP and then try *HELP Commands. These commands can be used not only directly from within the BASIC environment but also from within a BASIC program. In all versions of BBC BASIC (after BASIC I) these *-commands can be executed with the BASIC 'OSCLI' command. OSCLI expects a string variable which it then passes to the OS Command Line Interpreter, so you could use A$="HELP" : OSCLI A$. The use of OSCLI is illustrated by the BASIC program EmulateOS below.

Documented and Undocumented Calls

From the beginnings of the BBC range of computers in 1980, Acorn have provided a list of OS calls which they undertook to support on later versions of the machine. Some programmers have used 'short cuts' or undocumented calls, perhaps for speed, and then found that their programs would not work on upgraded machines such as the Master. My advice is not to use undocumented calls or 'clever' tricks that make assumptions about the use made by the OS of the machine's memory (e.g. where the character definitions are or where screen memory is). If you do, then keep that part of the program in a separate procedure and be prepared to rewrite it when the machine is upgraded!

The *FX Commands

On the earliest machines Acorn introduced the idea of using OS calls from within the BASIC environment. The location, and even the machine code, has changed with the model but, for example, *FX 12,3 still speeds up the auto repeat rate of the keyboard as it did on the very first BBC Micro. In my copy of the Archimedes User Guide the *FX commands are documented in chapter 24. *(Issue 3 User Guide has no chapter numbers! Try page 399. Ed.)*

The example program 'Screens' uses OS commands to swop between four screens. When you have played with the program as it is written, and if you have four favourite pictures called 'Myscreen1' etc. you can *SCREENLOAD them at say line 9160 with OSCLI ("SCREENLOAD Myscreen"+write$). In PROCerror I have included the commands to restore cursor editing (*FX 4), delay and repeat rates (*FX 12) as well as the default screen write and display (*FX 112 and 113). It is bad programming not to leave things as you found them (see lines 9260 and 9270).

OS_Word and Similar Commands

The memory of the original BBC Micro extends from &0000 to &FFFF (in 'hex'). Acorn reserved the top 'page' from &FF00 to &FFFF for, amongst other things, vectoring the OS calls. In other words, no matter where the actual routines were in memory, you called these addresses and they would 'point' to where the routine actually was. For the sake of upward compatibility, any calls to these 'magic' addresses on the Archimedes still 'work' as before. For example the *FX calls (otherwise known as OS_Byte calls) can be accessed through address &FFF4. Try the following:- A%=12:X%=1:CALL &FFF4. You will get the same effect as '*FX 12,1', so beware of assembling your own code to Archimedes locations &FF00 to &FFFF! On the old machines, OS_Word was called through &FFF1. This too still 'works' on the Archimedes. The original OS_Word calls were extended when the tape filing system was replaced by disc and there is now an OS_Word call which is equivalent to *FORMAT so be very careful not to CALL &FFF1!

The New Archimedes OS Calls

As well as maintaining a high degree of upward compatibility with the earlier BASIC/OS interface, Acorn have introduced a new way of calling OS routines. This is with the BASIC command SYS. These SYS commands are not listed in the User Guide but many are in the Programmer's Reference Manual. They can all be listed by running Neil Strong's program given in Volume 1, Issue 3 of Archive. He says there are 20 or 30 pages of calls: I believe him! In line 1040 of the program 'EmulateOS' replace the command OSCLI(command$) with SYS "OS_CLI",command$ and you will get the same effect as before. Now try replacing 9150 of 'Screens' with SYS "OS_Byte", 112,write%. Note that 'Byte' must have only one capital letter and be sure to use write%, not write$.

The main advantage of SYS (apart from the fact that it takes numeric variables and not just string variables) is that values can be returned to the user. For example, if you want to know the RS423 status byte (see programs for file transfer via RS423 port from Issue 1 and onwards of Archive) you can use SYS "OS_Byte",&9C ,0,&FF TO ,byte% : PRINT "&"; ~byte%. This value should be &56 for compatibility with the BBC Micro in its default state. The word TO allows information to be returned: the commas (particularly the one just before byte%) are most important. What it means is that byte% is the second variable returned by the routine. It is equivalent to saying 'TO x%,byte%' but because you are not using x%, you can just omit it. You can change the value of the status byte to, say, &52 by using the command SYS "OS_Byte",&9C,&52,0 TO ,oldbyte% : PRINT "&"; ~oldbyte%.

Further Expansion

The possibility for further expansion of both *-commands and SYS exists. The route is by creating relocatable modules. I have shown elsewhere in this issue how to create the new command "*MiniDump" (an Epson printer screen dump). There are facilities for introducing your own SYS calls as part of a module. Software developers should apply to Acorn for their own OS_name and the SWI number if they want to do this.

When NOT to use an OS Call

If you write programs in BASIC on the Archimedes and want them to run on another type of machine, for example an IBM in the 'Basic-86' environment, or on a Master fitted with 'Archimedes BASIC' then (I suppose it goes without saying) you must use only those calls that are supported by the environment. Since the BBC BASIC language is so well acclaimed, many machines (including RML) *(and the Mac!)* now have programs which will emulate, at least to some extent, BBC BASIC. OS_Byte calls particularly, which have three ways of being called on the Archimedes, may have only one way (usually with *FX) of being called on a non-Acorn machine. If you can obtain an effect by using a standard BASIC command then this may be slower but it will give more downward compatibility than the equivalent OS call. Finally, Acorn have suggested that, although the FX syntax supports commas (e.g. *FX &9C,&56,0) programmers should begin to use spaces instead (i.e. *FX &9C &56 0). There is always a reason why Acorn say things, but sometimes the reasons are to be found in the far future. *(I'm afraid I put commas back into all Gerald's FX calls because, with proportionally spaced text like this, commas are easier to see than spaces. Ed.)*

Summary

Programs are written in languages but executed in an environment. Over the range of BBC computers there have been four versions of BASIC and three major disc filing systems, (DFS with the 8271 FDC, ADFS with the 1770 FDC and Archimedes ADFS with 640/800k formats). All these (but particularly the disc filing system variants) are different environments and have caused many problems for users and software houses. Acorn have maintained a very high standard of upward compatibility and the Archimedes is no exception. Users writing there own programs are advised by Acorn, as well as me, to use only documented calls and no 'clever' memory oriented tricks (such as *SAVEing a screen) if they want their programs to run after an upgrade. The more I see of the Archimedes the more I feel confident that it has the upgrade potential for the 1990's. I wish you all Happy Computing in 1988 and painless upgrading in the years to come!

P.S. About Input/Output

I have now received a podule backplane and input/output podule but, as yet, no chips to upgrade from OS 0.2. Because things may be different with the latest OS, the following comments are only provisional.

I have a number of commercial add-ons for the old BBC Micro which plug into the User Port. These worked quite happily on the Master. Most of these add-ons need programs that are supplied as ROMs or in 6502 code but some have programs written in BASIC II. I have tried some of these BASIC programs with the user port of the I/O podule and none of them have worked first time. Looking at the programs, the reason is that the programs 'peek' and 'poke' values directly from or to the I/O memory area as used on the BBC Micro (quaintly referred to as SHEILA) with commands such as '?(&FE62)=&FF' (– a command to poke &FF into a register on the Interface Adaptor chip).

After changing the 'illegal' peeks and pokes to the corresponding OS_Byte calls I found the programs all worked well. Unfortunately it is not the habit of software houses to supply the source code of their machine code programs and it takes a lot of time and concentration to disassemble 6502 code (particularly if it is 'badly' written). The few ROMs I have looked at all use the 'illegal' peek and poke commands, probably for speed. Because of this, even with the 65Arthur emulator module installed, they fail to work on the Archimedes.

The I/O podule contains a 1MHz Bus, which is accessed through FRED (OS_Bytes &92 and &93) and JIM (&94 and &95), the User Port SHEILA (&96 and &97), the Analogue to Digital Converter (four more OS_Bytes or ADVAL in BASIC) and the 'hooks' for a MIDI interface. The moral of this story is "Don't use 'illegal' calls. They may not work when you upgrade your Archimedes!".

```
 100 REM > EmulateOS
 110 REM G L Fitton
 120 REM © ABACUS TRAINING
 130 REM Version 1.01
 140 REM 27th January 1988
 160 REM To demonstrate the use
 170 REM of OSCLI to emulate
 180 REM the OS environment
 190 REM from within BASIC.
 200 :
 210 ON ERROR PROCerror:END
 220 MODE 0
 230 :
1000 quit%=FALSE
1010 REPEAT
1020   INPUT'"Command="command$
1030   PRINT
1040   IF LEFT$(command$,1)="*"
     THEN OSCLI(command$) ELSE
     quit%=TRUE
1050 UNTIL quit%=TRUE
1070 END
1080
9000 DEF PROCerror
9010 *FX   4 0
9020 *FX  12 0
9030 *FX 112 0
9040 *FX 113 0
9050 VDU 26,12
9060 REPORT
9070 PRINT " at line ";ERL
9080 ENDPROC
```

```
 100 REM > Screens
 110 REM G L Fitton
 120 REM © ABACUS TRAINING
 130 REM Version 1.00
 140 REM 27th December 1987
 160 REM To demonstrate the use
 170 REM of OS commands
 180 REM to write and display
 190 REM different screenbanks.
 200 :
 210 ON ERROR PROCerror : END
 220 MODE 0
 230 :
1000 REM Core Section
1020 PROCscreens
1030 :
1070 END
1080
9000 DEF PROCerror
9010 *FX   4 0
9020 *FX  12 0
9030 *FX 112 0
9040 *FX 113 0
9050 VDU 26,12
9060 REPORT
9070 PRINT " at line ";ERL
9080 ENDPROC
9090
9100 DEF PROCscreens
9110 REM Writes to 4 different
                         screens.
9120 LOCAL write%,write$,
        display%,display$,quit%
9130 FOR write%=1 TO 4
9140   write$=STR$(write%)
9150   OSCLI("FX 112 "+write$)
9160   CLS
```

Automatic Re-Configuration

by Adrian Look

Have you got a program which needs a certain specific memory configuration? Just imagine, when all the software becomes available for the Arc, how difficult it will be trying to remember all the different set-ups! Well here's a way of making things a little easier.

Program 1 is a small ARM code program that causes the computer to reset itself – that is, simulate the user pressing the <break> key. It does this by setting the hardware reset vector to the appropriate value (as stored in location &3800000), entering into the supervisor mode, and finally running the reset code.

When you run program 1, it will produce an executable object code. That is the assembled version of the program. This is then saved on the disk as 'mreset'. Thus the reset may be executed by *RUNning this file.

Program 2 does exactly that. But before it does, it sets up the application's required configuration. This is done in PROCconfigure_machine (the values in this procedure are examples and you should replace them with your own).

For the configuration to take effect, the computer must cause a hard-reset. This causes the memory to be cleared and re-allocated. This is done by using the *FX 200,2 command.

The program also configures the machine to auto-boot mode, so that when it does cause a reset the machine will not only be in the new configuration but it will re-boot the program disk, thus running program 2 again, which is why it must be the !BOOT program.

When run for the second time, it senses this auto-booting and runs the application, but not before turning the auto-booting back off again.

So here's what to do:

1) RUN program 1, with your program disk in the disk drive. This will then save '$.mreset' on the disk.

2) Alter program 2 so that it configures the machine to what you want and, also replace the '..main.program.filename..' with the name of the program to be run by the !BOOT file.

3) SAVE the altered program 2 on the same disk as the !BOOT file.

```
9170   PRINT TAB(0,write%*4);"Screen bank ";write$;"."
9180   PRINT "Choose a screen to view between 0 and 4."
9185   PRINT "Press Q to quit."
9190 NEXT write%
9200 REPEAT
9210   REPEAT
9220     display$=GET$
9230   UNTIL INSTR("01234Qq",display$)
9240   IF INSTR("01234",display$) THEN OSCLI("FX 113 "+display$)
9250 UNTIL INSTR("Qq",display$)
9260 *FX 112 0
9270 *FX 113 0
9280 ENDPROC
```

4) Finally, set the disk's boot option to RUN. This is done by typing: *OPT 4,2 <return>.

As a final note, Acorn strongly recommend that configuration of the machine should be left to the users themselves! So if you intend to sell any software that requires special configurations, I suggest that you install such a program but also allow the user to 'turn it off' if he/she so desires. That way, everybody is happy??

Program 1

```
 10 REM >$.Reset
 20 REM Resetting your machine
 30 REM Adrian Philip Look
 40 REM 20th January 1988
 50
 60 DIM code% &FF
 70 FOR opt=0 TO 3 STEP 2
 80 P%=code%
 90 [OPT opt
100
110 MOV R0,#&3800000
120 LDR R1,[R0]
130 STR R1,[R1,-R1]
140 SWI &16
150 TEQP PC,#&FC000003
160 MOV PC,#0
170 ]
180 NEXT opt
190 OSCLI ("SAVE $.mreset "
     +STR$~(code%)+" "
     +STR$~(P%))
200 END
```

Program 2 :-

```
 10 REM >$.!BOOT
 40 REMConfiguring your machine
 50 REM Adrian Philip Look
 60 REM 20th January 1988
 80
 90 *fx 200,2
100 SYS "OS_Byte",161,16 TO
                      ,,misc_flags
110 auto_boot=(misc_flags AND
                           %10000)
120
130 IF auto_boot=0 THEN
140 PROCconfigure_machine
150 file$="$.mreset"
160 *configure boot
170 ELSE
180 file$="main.prog.filename"
190 *configure noboot
200 ENDIF
210
220 OSCLI(file$)
230 END
240
250 DEFPROCconfigure_machine
260 *configure ScreenSize 0
270 *configure RMASize 0
280 *configure SpriteSize 2
290 *configure RamFsSize 0
300 *configure SystemSize 0
310 *configure FontSize 0
320 ENDPROC
```

Program Listings

You will find that in a number of the program listings in this issue, the numbering is not consecutive in places. This does not mean, I hope, that certain vital lines of the program are missing. What I have done is to miss out some of the blank lines that the programmers had put in to aid clarity. I have done so because I am trying to fit a quart's worth of programs and articles into a pint sized Archive pot! Actually, I have increased the pot size by four pages but it was still a bit of a squeeze to get it all in. I hope this has not spoilt the normal clarity of the magazine.

Painting on the Archimedes

A comparative review of Arctist and Artisan by Malcolm Banthorpe

Why paint?

"Arctist" from Fairhurst Instruments Ltd. and Clare's "Artisan" are among the first and, as yet, few third party software releases for the Archimedes. It is therefore appropriate, given the machine's superior graphics capabilities, that they should be "painting" programs. By "painting", I mean that they are graphics programs whose aim is not to produce technical drawings and diagrams in the way that CAD packages do, but to to give aspiring computer artists full reign to their creativity.

A lot of people who would never dream of touching a paintbrush (apart from applying a coat of magnolia to the spare bedroom) or of putting pencil to paper in order to produce a picture for its own sake, are nevertheless attracted to computer painting programs. This attraction may sometimes stem from the vague hope that the power of the computer will magically compensate in MIPs. for what they lack in manual skills and experience. Sadly this cannot be so; it takes at least as much skill and experience to produce a masterpiece on a screen as it does to do the same in, say, water-colour. However, computer paint packages do offer the opportunity to produce an acceptable, even pleasing, result with a rather smaller investment of time and effort than is required in other media. The situation, perhaps, has a parallel in the field of music in that most people can, with a little practise, produce some musically acceptable sounds from a modern electronic keyboard. To produce an equally acceptable sound from a violin takes a considerable amount of practise and dedication. To produce great music from either requires many years of practice and dedication.

Computer graphics, at least in the form presently accessible to most of us, imposes restrictions in colour and spatial definition when compared with what is possible with even a child's paintbox and a piece of paper. So from one point of view, coupled with the fact that a mouse is generally less suited to freehand work than a pencil or brush, we start at a disadvantage. On the other hand there are a whole lot of things that you can do to manipulate an image on a screen that would be either extremely tedious or, quite often, just plain impossible on paper.

Computer paint programs are attractive because of the way they can manipulate images and can make easy work of such tasks as drawing straight lines, circles and smooth curves, and of precisely filling areas with a uniform colour or pattern, which are time consuming with pencil and paper. Moreover they allow mistakes to be quickly rectified and offer number of unique facilities such a instant cut and paste.

What software?

Both programs reviewed here are mouse driven, allowing the whole screen to be used for a picture and make extensive use of icons to symbolize the various drawing functions. "Arctist" offers the use of 256 colours simultaneously on the screen; "Artisan" is restricted to a palette of 16, but each of these may be selected from the Archimedes total range of 4096 colours.

"Arctist" is designed for use with the A310, A410 or A440 whereas "Artisan" will work with any Archimedes model. Using "Artisan" on a machine with 512k of memory will place some restrictions on the number and size of sprites which can be held in memory. In this context the word, sprite, as you're probably aware, refers not to a small space invader figure to be used in a game but to any rectangular area of the screen, up to its full size, which is stored in memory or on disc for future use.

Arctist?

Of the two programs reviewed, "Arctist" is the simpler – and also the cheaper at £19.95. The copy received was a pre-release version and lacked the instruction manual that will accompany the final version. Nevertheless the the icon-driven program was easy to use and the lack of explicit instructions posed no problems. The menu always appears across the top of the screen, with the left-hand mouse button being used to select a particular function. The middle button toggles the menu on and off; the right-hand button de-selects or cancels a function.

If the colour choice icon is selected, the main menu is replaced with a sub-menu showing 3 rows of 23 different colours, each in four different tints. Hang on.... 3 times 23 times 4 makes 276; the Archimedes can display a maximum of 256 colours simultaneously; clearly something is not quite right here. Closer inspection reveals that the 20 extra colours are accounted for by 5 of the colours and their tints being repeated. Why this has been done is mystery. Admittedly the actual number of colours available, 64, would not give 3 equal rows but duplication of some of the colours seems a potentially confusing way to get round a screen layout problem – particularly when many of the colours appear similar when seen in this context.

The other options on the main menu are pretty well as would be expected from a paint program. Rubber-banded lines, rectangles, ellipses and triangles are available. Shapes may be either outline only or filled. There is no separate circle function – it's up to you to set an ellipse with equal major and minor axes. The fill option works only with solid colours; patterned fills are not catered for within the program.

Any selected screen rectangle can be flipped horizontally or vertically or have its colour inverted. Rectangular sections can also be grabbed as sprites and hence copied elsewhere on the screen.

A spray function allows airbrush effects, with 10 different spray intensities selectable from a sub-menu. The spray pattern is square rather than the more desirable circular shape.

Text may be positioned anywhere on the screen and placed to the nearest pixel but unfortunately only in the system's 80 column font, which may be too small for some display purposes. Another minor niggle is that the DELETE key doesn't do anything. If you make a typing error, the general rub-out function needs to be called to erase the appropriate section of screen. If RETURN is pressed while typing text then the cursor will be placed immediately below the original starting point, making it easy to align a column of text.

A pixel-editing function is provided by displaying a magnified version of a selected 16 x 16 pixel section of the screen, at its top right-hand corner. That means that the area covered by the magnified section is not available for editing, but this is simply overcome by providing a scroll function whereby the whole screen can be shifted to the left. The bit that disappears off the left-hand edge wraps round to the right. After editing, a further scroll to the left will eventually re-position the screen as it started. Other functions allow rays to be drawn from a point and the screen to be cleared – but only to black; if you want a different background colour the only way is to 'fill' the whole screen with it.

Disk access is also provided within the program so that screens and sprites can be saved and loaded.

To quote from the blurb on the disc envelope :

"The need was to be able to design screens in 256 colours that could then be used in games, title pages etc With full screen editing, zoom & the ability to save an area of screen as a sprite. Yet easy to use within a very short time." (sic)

The program achieves those stated objectives but little else, so the program can be recommended to anyone whose requirements are for a simple 256 colour paint package, although I feel that most people who have spent their hard-earned cash on an Archimedes are looking for software that goes further in exploiting its power.

(The quotation above has been corrected in the manual on the full-release version, but the English is still not all it might be! Ed.)

Artisan?

"Artisan", at £39.95, costs almost twice as much and is limited to 16 colours, but has a lot more to offer. It is supplied with a 37 page manual which concisely explains all the functions available. Being one of those impetuous people who, against all advice, always plunges straight in without reading instruction manuals first, I booted up the disc as soon as it arrived. It was probably fortunate that one of the first icons (an exclamation mark) I selected was for HELP. This opens a window at the base of the screen and gives a sort of running commentary on what you are doing or should be doing at that stage. (Pointing at the help window itself results in the facetious prompt, "Go and point somewhere useful!") *(If you think that's facetious, you should have seen the pre-release version!! Ed.).*

As you move the mouse pointer over a series of menu icons, the help window is constantly updated to keep you informed of the function offered by each one. Thus I found it possible to make my way around what is a fairly complex program, referring only later to the manual, in order to pick up some of the finer points. After a short time, it's possible to become familiar enough with the icons to dispense with the Help window, but it's nice to know that it's always available to be toggled on and off as required.

The program is run from a series of pop-up menus. When an item has been selected, moving the pointer off the menu to makes it vanish. This is slightly disconcerting at first, as one false move in the wrong direction can make the menu disappear before you have selected anything. However, once I had got used to this I found it a very fast way to work, eliminating quite a lot of unnecessary button pushes. The middle mouse button can always be used to recall the last menu, the left-hand button is used to select a function. The main purpose of the right-hand button is to UNDO the last function. This very welcome facility means that if you have made a mistake since the last menu function was selected, such as a line in the wrong place or a fill that "leaked", then you can restore the screen as it was before the menu selection, simply by pressing the right-hand button.

The pop-up menus make proper use of the window manager and, like the accessories in the desktop, can be dragged to a new position on the screen. Once a particular menu has been re-positioned, it will appear there each time that it is called. Different menus may be in different positions.

The functions offered by the program are too numerous to describe in detail so I'll stick to brief mentions for all but the more unusual ones.

The Paint menu offers circular, square, triangular and elliptical brushes with which to draw in any of the solid colours or from extended colour fill patterns (referred to as ECF's within the program).The size of each brush can be varied from a single pixel upwards. Six patterns are already defined and available on the colour selection menu, other sets can be loaded from the system disc or can be defined at any time by the user. Other brushes can be defined within an 8 x 8 matrix and can contain any selection of colours. Any of the 16 colours within the standard palette can themselves be re-defined using red, blue and green sliders, as in the desktop program. Mouse pointer and border colours are similarly re-definable. Re-defined palettes may be stored on disk for future use. A zoom function magnifies part of the screen to almost full screen size for pixel editing. Magnification can be x2, x4 or x8. Again the window manager module is used to give a standard window as seen on the desktop and with the scroll bars used to pan over the picture.

Also on the Paint menu and deserving of special mention is a facility called Magic Brush. At first this appeared to me to be little more than a gimmick but subsequently proved itself to be quite a versatile tool. The brush itself is a square of definable size. When it is passed over a picture, all pixels of a pre-selected colour are changed to a second pre-selected colour. In addition, all pixels of a pre-selected colour may be replaced with alternate pixels of two other pre-selected colours. This goes a long way in compensating for the restrictions of the 16 colour mode. A whole new set of dithered colours is possible – in theory, 256 in all. For instance if there is an area of green in the picture which you think would look better if slightly darker, you simply set the Magic Brush palette to replace green with alternate green and grey pixels and then pass the Brush over the area.

The Banding menu offers circles, rectangles, triangles, ellipses, circles, parallelograms, lines, linked lines, segments and chords as rubber-banded shapes, either as outlines or filled. In addition, a user-defined shape of up to 64 lines may be memorised and the rubber-banded.

The Sprite menu, as the name suggests, is concerned with displaying sprites stored in memory, loaded from disc or grabbed from the screen. Worthy of note here is that any of the 16 colours can be declared as transparent so that although a sprite is stored as a rectangular block, a mask can also be stored and it can be displayed as any shape. In practice this means that if you draw a black cat on a yellow background, then declare white as transparent and grab the cat as a sprite, the cat can then be re-drawn on any background without the original yellow background being reproduced. This provides a true cut and paste facility, which is a considerable advance on that which has been available on most micros until recently. A sprite can also be used as a brush for drawing, which can lead to some interesting effects.

The Cut and Paste menu expands on the effects available from the sprite menu. Again, a colour can be declared as transparent. Thus to cut an object from a detailed background, first paint round the object, to obscure the background, with a colour that does not occur within the object. Next, declare that colour as transparent and then grab the rectangle containing the object. Objects thus grabbed can be flipped horizontally and vertically, scaled and rotated by any whole number of degrees. When objects are rotated there may be some loss of detail, particularly if they contain patterned fills. This is an unavoidable outcome of the fact that mode 12 pixels are not square. A possible remedy is to apply the fills after rotation. It's also a good idea to apply fills after rather than before scaling, when possible.

If "Artisan" is used on an A305 with 512k of ram there will be some restriction on the size of sprites; with 1 Megabyte, sprites may be up to full screen size.

The Toolkit menu contains a number of facilities that don't quite fit any of the other categories. Whole and partial toned screen dumps are available for Epson printers. (The system disc also contains a colour dump for Integrex printers). There is provision to adjust the mouse sensitivity, although I haven't yet encountered any situation where less than maximum sensitivity was desirable. The Clear Screen function appears on this menu and clears the screen to the currently selected colour or pattern. A rectangle of screen may be distorted to a pre-defined shape. Its use is restricted by the limited form of distortion offered but it can, amongst other things, achieve certain perspective effects and convert text in a standard font to italic. Text can be placed on the screen in any of 10 supplied fonts, all proportionally spaced and quite elegant (and DELETE does work). An unusual feature, at least for a paint program, is an edge detector. Within a selected area of screen, all areas of uniform colour are converted to white

Diagram II by Pineapple Software

A Review by Matthew Treagus

Diagram II is a drawing and design package (not an art package like Artisan or Arctist) for the Archimedes which has been converted from the BBC/Master program and put onto an 800k ADFS disk. The original BBC documentation has been supplemented with a photocopied sheet for Archimedes users. The program still runs in an 80 column, two colour mode so that the main application would be for preparing black-and-white artwork more quickly and neatly than could be done by hand. The only addition to the program seems to be the mouse option, though in many ways this is no great advantage since many commands still require the use of the keyboard.

The disk auto boots into a simple menu offering diagram creation and access. The diagrams are defined in screens. A diagram can be x screens wide and y screens high of which one screen is displayed at a time although a small screen option is available that condenses the image into one screen for overall viewing.

The actual diagram editor is rather primitive in some respects yet in others it excels itself by providing unbelievably useful services and facilities. Despite the sophistication of certain of the features, the basics such as shape drawing leave something to be desired. All preset shapes are selected with the control key plus a function key and the triangle and square/rectangle commands were left out, due to lack of space, it seems. What I could not understand is why there are only filled shapes available with the exception of Arcs, Ellipses and Circles. I appreciate they can be drawn with the "line-drawer" but surely it would be just as easy to provide a Fill/Outline toggle key and effectively double the number of shapes usable, as indeed Artisan does, or even give just outline shapes and flood fill them later.

A pixel editing mode is available at screen level only, i.e. it has no zoom facility, which makes it difficult to use. There is however a drawing mode which allows junctions automatically drawn – very useful for circuit diagrams.

But enough moans and groans, the package does have some very strong points which include the high quality printer dumps and UDC's (User Definable Characters). The printer driver appeared with an impressive list of options and

---

(Arctist/Artisan Review continued…)

and their edges to black. Again, a facility of limited use but it provides interesting variations on the fonts if applied to enlarged text.

Finally, the Disc Access menu allows saving and loading of sprites, patterns, palettes and screens and the system disc contains samples of each.

An "Artisan" utilities disc will be available containing more printer dumps, a "slide-show" facility to display a series of screens joined by either supplied or user-defined wipes, more fonts and more sprites. I also understand from Clares that a 256 colour version of "Artisan" is in preparation for release at a later date.

Summary

Both "Arctist" and "Artisan" are easy to use but vary considerably in the range of facilities offered. As far as I'm concerned "Artisan" wins hands down as a general painting package and is well worth the extra money. If you specifically need a basic full-screen painting program with 256 colours and pixel editing then "Arctist" is worth considering, as it performs well within its limits. Otherwise, if you require a simple package the free Paint program on the Welcome disc is worth close attention. "Artisan" goes considerably beyond either of these and is, I feel, just a taste of the sort of quality RISC software that we can hope to see in the future.

settings that should cater for almost every printer under the sun and these settings can then be saved as a print file which can be reloaded at a later date. These options included margins, rotation, scaling (important for PCB's) and paper width. The print quality was exceptional – my clapped out old GLP II with a dry ribbon produced extremely clear print and the output from my Kaga/Taxan was unbelievable.

The UDC's are used to the maximum by Diagram-II and four font sets are provided on the disk. Circuit symbols, musical notes, furniture etc, could be defined and incorporated into diagrams. The creation and editing of UDC's is aided by the mouse and for this I really found the mouse addition useful. There are 880 UDC's available to the user. Text files can be loaded onto the screen and formatted into any area.

The package gives the impression that it has rather been rushed into production, but then that's what the consumer wants – software NOW. There are some very strong and useful features but others which could have been further developed. The package is not altogether user-friendly despite the neat movement between menus. It has great potential, but at £63.25 inc VAT you would only buy it if you had a real practical use for it – it's not just a "fun" drawing package – it's for serious diagram drawing. I would be happier to pay that much money if there were a BASIC V or ARM code version available, but if you have lots of diagrams to draw than the time saved could well be worth the money. A

Jim Daniels of Pineapple Software would like us to point out that the emulator version of Diagram-II was mainly aimed at existing users (and there are apparently hundreds of them – many from large companies like British Aerospace) who are upgrading to Archimedes and who want to continue using Diagram-II. They in fact can get the upgrade free of charge.

The PC Emulator – A Review

by Mark Sealey

One aspect of the Archimedes system that has received as much attention in the computing press as almost any other is that of PC compatibility. Originally, Acorn had intended to produce a PC Podule. The word now is that the software under review here is felt to be perfectly adequate for those who want their machines to load and run software originally written for the MSDOS and/or IBM PC environment. Is it?

Well, the first thing that will strike you when you open the package is how slim and small it is! There are two discs, one in 800K ADFS and the other 720k IBM format, a registration and licence agreement to sign and an eleven page booklet. Yes eleven pages, four of which are all but blank!

Given that MSDOS (you get version 3.1 as well as GW Basic) is quite unlike ADFS and that the filenames, for instance, have an entirely different appearance with their three letter extensions after a dot, such scant documentation is amazing. I would go so far as to say that anyone new to MSDOS would be practically unable to use the package without one of the guides or books mentioned at the back of this booklet.

Even then, the two utilities (PUTFILE.EXE and GETFILE.EXE) intended for MSDOD-ADFS transfer and vice-versa would nowhere be mentioned. Nor would the fact that to echo screen output to the printer CTRL-B and C no longer work; this is the true MSDOS environment: it's toggle CTRL-P or nothing. Not even Control-Break works. Use the reset button.

At this point, it should be said that the environment appears quite sound. Key f3 recalls the last command entered at the keyboard, ALT-CTRL-DELETE performs a soft reset for rebooting and so on. This is all in the Emulator's favour – though not in its manual. If you are used to working with MSDOS on machines like the IBM and Nimbus, you'll quickly get over the strangeness and feel very much at home.

So the product is very much a mixture. On the one hand, the emulation seems to be very good; on the other, if you haven't used MSDOS before, you'll need much more help than the package you've just spent over £100 on can give you.

How complete is the emulation? I could load Wordstar, but not run it properly; the same for Word Perfect. None of the graphics programs currently used in schools, such as PaintSPA, PC Paintbrush or NewSPAper would work. Lotus 123 and dBase III apparently do run. Yet in the absence of really detailed documentation, it could well be that machine configuration and memory allocation expected by the Emulator had something to do with this......who knows?

What is needed is the compilation of a list of software that will work – similar to that pioneered by Tubelink and Benjamin Rietti for those BBC ROMs that would work across the Tube. Any takers? It goes without saying that unless your software will run, there's little point in having this package!

Disc handling itself is smooth. The display is in an IBM-style font in toned-down grey and really runs quite slowly. Yet probably no more so than many MSDOS machines themselves, which are 8 to 16 bit. The fact that my Archimedes is loading programs form discs formatted and intended for a Nimbus is itself an achievement. The drives even sound like a Nimbus, rather than the way they usually do. Furthermore, it is possible to have a partition of your Winchester in MSDOS format.

Conclusions

When you consider who will be interested in the package, the position becomes clearer. It is doubtful if anyone will buy a 310 and use it to run a 'dBase' with products like Minera's 'System Delta Plus' around. Nor – with clones such as Amstrad's costing less than £400 – will many people be buying Archimedes as a business machine to compete with their PCDOS and MSDOS-based ones.

The market must surely be commercial and education users who want to standardise on Acorn equipment for its other outstanding features and at the same time extend its scope to cover the world of MSDOS software. Until we know just what will run and what won't with greater certainty, it's difficult to know whether their needs will be met by the emulator. If it becomes obvious that they can just "drop in" packages such as LTS 'News Master', as is rumoured, then it will quickly pay for itself.

As for schools and colleges, many of whom will need the use of Graphics as standard, it is clear that in some cases they will be disappointed. Either because Enhanced Graphics just will not work or, as I suspect is the case with the programs above, 640k free is needed rather than the 360k which the Emulator gives you on an Archimedes with only(!) 1Mbyte of ram.

As an exercise in total emulation, this would seem to be a success. As an overall product at £113.85 with such a stunning lack of documentation, it leaves something to be desired. A

Has anyone tried to contact LTS as mentioned last month? We can't get any response.

Word Perfect doesn't run on the version of the emulator that Mark was reviewing due to a bug. The good news is that there is a version of the emulator that ***does*** *works and is about* ***40% faster*** *than before. The bad news is you have to send Acorn £15 plus the old disc to get it!*

3-D Button Effects

Rob Hindle

This procedure is designed to create an effect similar to that seen in some of Clares' Archimedes software products. The icons used to represent different system functions give the impression of being in rectangular recesses on a 3-dimensional surface. The procedure is quite simple, it requires three colours to be defined being three shades of the same basic colour. Two adjacent triangles are drawn to make a rectangle, one is in the palest shade and the other in the darkest. The orientation of these triangles indicates the direction of the light source, generally if the light comes from above, the button will appear to be recessed, if from below it appears to stand up from the screen. Finally a rectangle is drawn on top of the two triangles but smaller than them so their edges remain as a border, the rectangle being in the middle shade of the three colours.

There are two sample programs which use the same buttons procedure. The first is a fuller demonstration of its capabilities – some of the buttons are dynamic and represent which function keys have been pressed. Some extra features have been used to enhance the 3-D effect, the lettering on the keys is moved slightly when the key is pressed, the colours used for the recessed buttons are slightly darker than normal and the keytop is drawn a little smaller. The second demo is short to demonstrate the general idea with a minimum of typing.

The parameters used by PROCbuttons are:

colour% – a number in the range 0 to 3 allowing 4 different colours of button to be used. You need to define these as colours 4 to 15, in 4 groups of 3 shades.
X%, Y% – bottom left co-ordinates of button
wide%, high% – button width and height
orientation% – direction from which the illumination appears to come, values {0-3}.

border – this is a fractional value of less than 1, it indicates the percentage of the button which will be used for the border around the keytop. In general this needs to be smaller for larger buttons, see the examples.

```
 10 REM >BUTTONS
 20 MODE12
 30 VDU5
 40 *FX225,2
 50 *FX221,2
 60 DIM fnkey%(12)
 70 fnkey%()=0:fnkeys$=CHR$&FF
 80 FOR X%=&81 TO &89
 90   fnkeys$+=CHR$X%
100 NEXT
110 fnkeys$+=CHR$&CA+CHR$&CB+
                         CHR$&CC
120 COLOUR3,255,255,255
130 COLOUR4,112,80,64:COLOUR5,
      64,80,64:COLOUR6,144,80,64
140 COLOUR7,0,128,0:COLOUR8,0,
            80,0:COLOUR9,0,160,0
150 COLOUR10,128,80,64:COLOUR11
     ,80,80,64:COLOUR12,160,80,64
160 COLOUR13,0,0,160:COLOUR14,
          0,0,80:COLOUR15,0,0,208
170 GCOL 135:CLG
180 PROCbutton(1,20,16,1239,
                     991,0,.02)
190 FOR X%=0 TO 11
200   PROCbutton(2,X%*98+60,64,
                    80,64,0,.15)
210   MOVE X%*98+72,104
220   GCOL 0
230   PRINT"F";STR$(X%+1)
240 NEXT
250 PROCbutton(3,340,512,600,
                     384,1,.05)
260 PROCbutton(3,440,616,400,
                     172,0,.05)
270 GCOL 3
280 MOVE 450,840
290 PRINT"3-D Button
                demonstration"
```

```
300 MOVE 470,720
310 PRINT"Press a function key"
320 REPEAT
330   REPEAT
340     X$=GET$
350     X%=INSTR(fnkeys$,X$)-2
360   UNTIL X%>=0
370   fnkey%(X%)=ABS(fnkey%(X%)-1)
380   IF fnkey%(X%)=1 THEN
390     PROCbutton(0,X%*98+60,
                64,80,64,1,.20)
400     MOVE X%*98+72,108
410     ELSE
420     PROCbutton(2,X%*98+60,
                64,80,64,0,.15)
430     MOVE X%*98+72,104
440   ENDIF
450   GCOL 0
460   PRINT"F";STR$(X%+1)
470 UNTIL FALSE
480 END
490
500 DEFPROCbutton(colour%,X%,Y%
,wide%,high%,orientation%,border)
510 GCOL6+3*colour%
520 xBorder=border*wide%
530 yBorder=border*high%
540 CASE orientation% OF
550   WHEN 0:MOVEX%,Y%+high%
     :MOVEX%+wide%,Y%+high%:WAIT
   :PLOT85,X%,Y%:GCOL5+3*colour%
             :PLOT85,X%+wide%,Y%
560   WHEN 1:MOVEX%+wide%,Y%
      :MOVEX%,Y%:WAIT:PLOT85,X%+
  wide%,Y%+high%:GCOL5+3*colour%
             :PLOT85,X%,Y%+high%
570   WHEN 2:MOVEX%+wide%,Y%+
      high%:MOVEX%,Y%+high%:WAIT
     :PLOT85,X%+wide%,Y%:GCOL5+3
          *colour%:PLOT85,X%,Y%
580   WHEN 3:MOVEX%,Y%:MOVEX%
     +wide%,Y%:WAIT:PLOT85,X%,Y%
         +high%:GCOL5+3*colour%
       :PLOT85,X%+wide%,Y%+high%
590 ENDCASE
600 GCOL4+3*colour%:RECTANGLEFILL
     X%+xBorder,Y%+yBorder,wide%-
        xBorder*2,high%-yBorder*2
610 ENDPROC
```

```
 10 REM >BUTTONS2
 20 MODE12
 30 COLOUR13,128,128,128
              :COLOUR14,96,96,96
           :COLOUR15,160,160,160
 40 GCOL 141:CLG
 50 FOR X%=0 TO 11
 60   PROCbutton(3,X%*98+60
               ,160,80,64,1,.15)
 70   PROCbutton(3,X%*98+60,64
                  ,80,64,0,.15)
 80 NEXT
 90 END
100
110 DEFPROCbutton(colour%,X%,Y%
       ,wide%,high%,orientation%
                        ,border)
120 GCOL6+3*colour%
130 xBorder=border*wide%
140 yBorder=border*high%
150 CASE orientation% OF
160   WHEN 0:MOVEX%,Y%+high%
     :MOVEX%+wide%,Y%+high%:WAIT
          :PLOT85,X%,Y%:GCOL5+3*
     colour%:PLOT85,X%+wide%,Y%
170   WHEN 1:MOVEX%+wide%,Y%
       :MOVEX%,Y%:WAIT:PLOT85,X%
         +wide%,Y%+high%:GCOL5+3
     *colour%:PLOT85,X%,Y%+high%
180   WHEN 2:MOVEX%+wide%,Y%+
      high%:MOVEX%,Y%+high%:WAIT
     :PLOT85,X%+wide%,Y%:GCOL5+3
           *colour%:PLOT85,X%,Y%
190   WHEN 3:MOVEX%,Y%:MOVEX%+
      wide%,Y%:WAIT:PLOT85,X%,Y%
          +high%:GCOL5+3*colour%
       :PLOT85,X%+wide%,Y%+high%
200 ENDCASE
210 GCOL4+3*colour%:RECTANGLEFILL
     X%+xBorder,Y%+yBorder,wide%
       -xBorder*2,high%-yBorder*2
220 ENDPROC
```

Design Your Own Mouse Pointer

Keith McAlpine

Inspired by Jeff Shipp's "Hand" pointer, Keith has written a program so that you can design your own pointers.

The DefPointer program is used for designing the mouse pointer whereas the SetPointer program (Jeff Shipp's Hand program with a few modifications) actually displays the pointer.

On running the DefPointer program you are presented with a screen as shown opposite.

On startup, the pointer shape is a maximum size (8 x 32) and the active position is the top left-hand corner, with red as the current brush colour.

To change colour, move over the colour box wanted and press <select>.

To redefine the current colour, move over the colour circle using <menu> and <adjust> buttons to vary the colour intensities.

To plot a point, move over the grid and press <select>.

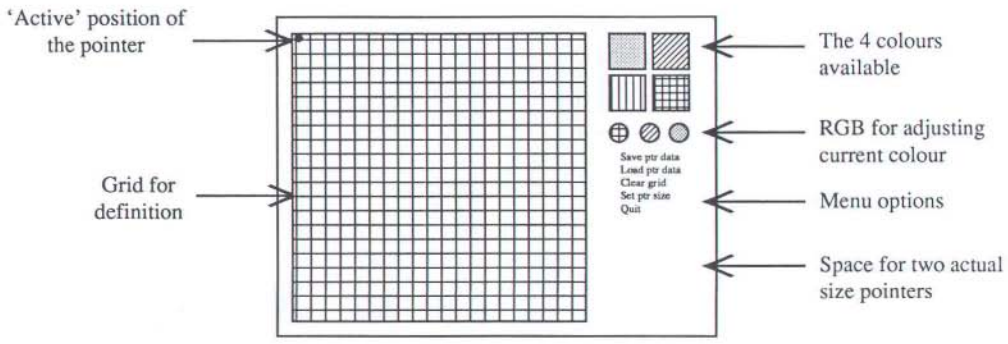
To move the active point, move over the grid and press <menu>.

To select and item from the options table, move over the table and a "*" will appear at the current selection. Press <adjust> to select it.

All of the options are self-explanatory except "set ptr size". After selecting this option, part of the grid is inverted. Move over the grid, holding down the <select> button. The inverted area is the area that will be kept. Press <menu> to select your chosen inverted area. Have fun with it!

```
 10 REM >DefPointer
 20 REM Program for changing the pointer
 30 REM by Keith McAlpine & Gem Electronics
 40
 50 MODE12:OFF
 60 DIM cols(2,2)
 70 cols(0,0)=240:cols(0,1)=0:cols(0,2)=0
 80 cols(1,0)=0:cols(1,1)=240:cols(1,2)=0
 90 cols(2,0)=240:cols(2,1)=240:cols(2,2)=0
100 FOR lp=0 TO2:COLOUR lp+1,cols(lp,0),cols(lp,1),cols(lp,2):NEXT
110 VDU23,224,255,128,128,128,128,128,128,255,23,225,255,1,1,1,1,1,
                                                                 1,255
120 COLOUR 13,255,0,0:COLOUR 14,0,255,0:COLOUR 15,0,0,255:COLOUR 7,
130 PROCdrawgrid                                           128,128,128
140 GCOL0:RECTANGLE FILL 1050,900,92,92
150 GCOL1:RECTANGLE FILL 1150,900,92,92
160 GCOL2:RECTANGLE FILL 1050,800,92,92
170 GCOL3:RECTANGLE FILL 1150,800,92,92
180 FOR lp=13 TO 15:GCOL lp:CIRCLE FILL (lp*80)+24,740,24:NEXT
190 COLOUR6:PRINTTAB(66,12)"Save ptr data"TAB(66,13)"Load ptr data"
200 PRINTTAB(66,14)"Clear grid"TAB(66,15)"Set ptr size"TAB(66,16)
                                                                "Quit"
210 *POINTER
220 col%=1:ny%=0
230 REPEAT
240   MOUSE x%,y%,b%
250   IF x%<1024 THEN PROCmaingrid
```

```
260   IF x%>1049 AND y%>799 AND b%=4:col%=POINT(x%,y%):SOUND 1,-15,
                                                        (col%*48)+48,3
270   IF (x%>=1040 AND y%>700 AND y%<770) AND b%<4 THEN
280     paracol=POINT(x%,y%)-13:IF paracol<0 OR col%=0 THEN 320
290     IF b%=1 AND cols(col%-1,paracol)>0 THEN cols(col%-1,
                   paracol)-=1:SOUND 1,-15,cols(col%-1,paracol),8
300     IF b%=2 AND cols(col%-1,paracol)<240 THEN cols(col%-1,
              paracol)+=1:SOUND 1,-15,cols(col%-1,paracol)/16*4+50,5
310     COLOUR col%,cols(col%-1,0),cols(col%-1,1),cols(col%-1,2)
320   ENDIF
330   X%=x% DIV 16:Y%=(1023-y%) DIV32
340   IF X%>64 AND Y%>11 AND Y%<17 THEN COLOUR 5:VDU31,65,ny%,32
                  :ny%=Y%:VDU31,65,ny%,42:IF b%=1 THEN PROCoptions
350 UNTIL 0
360
370 DEFPROCoptions
380 SOUND1,-15,150,3
390 CASE ny% OF
400   WHEN 12:PROCsave
410   WHEN 13:PROCload
420   WHEN 14:PROCdrawgrid
430   WHEN 15:PROCsetsize
440   WHEN 16:MODE0:PRINT'"Finished"'':END
450 ENDCASE
460 SOUND1,-15,80,3
470 ENDPROC
480
490 DEFPROCsave
500 file$=FNfilename
510 GCOL3,7:CIRCLE FILL (mx%*32)+16,((31-my%)*32)+16,12
520 file=OPENOUT(file$)
530 BPUT#file,0:BPUT#file,2:BPUT#file,sx%:BPUT#file,sy%:BPUT#file,mx%
```



```
540 BPUT#file,my%:BPUT#file,0:BPUT#file,0:BPUT#file,0:BPUT#file,0
550 FOR lp1=0TO2:FOR lp2=0 TO2:BPUT#file,cols(lp1,lp2)
551 NEXT:NEXT
560 FOR y%=1010 TO (31-sy%)*32 STEP-32:FOR x%=16 TO sx%*128 STEP 128
570     BPUT#file,FNbyte(x%,y%)
571 NEXT:NEXT:CLOSE#file
572 GCOL3,7:CIRCLE FILL (mx%*32)+16,((31-my%)*32)+16,12
590 PRINTTAB(64,10);SPC15
600 ENDPROC
610
620 DEFPROCload
630 file$=FNfilename:file=OPENIN(file$):CLOSE#file:IF file=0 THEN
                                                          VDU7:GOTO770
640 GCOL3,7:CIRCLE FILL (mx%*32)+16,((31-my%)*32)+16,12
650 file=OPENIN(file$)
660 VDU28,66,30,78,18,12,28,0,31,64,0,12,26
670 d=BGET#file:d=BGET#file:sx%=BGET#file:sy%=BGET#file
                                                       :mx%=BGET#file
680 my%=BGET#file:d=BGET#file:d=BGET#file:d=BGET#file:d=BGET#file
690 FOR lp1=0TO2:FOR lp2=0TO2:cols(lp1,lp2)=BGET#file
691 NEXT:NEXT
700 FORlp=0TO2:COLOUR lp+1,cols(lp,0),cols(lp,1),cols(lp,2):NEXT
710 FOR y%=0 TO sy%-1:FOR x%=0 TO sx%*8-1 STEP 8
720   byte%=BGET#file:p1=(byte% AND 3):p2=(byte% AND 12)>>2
730   p3=(byte% AND 48)>>4:p4=(byte%AND192)>>6:PROCplot(x%+6,y%,p4)
740   PROCplot(x%,y%,p1):PROCplot(x%+2,y%,p2):PROCplot(x%+4,y%,p3)
750 NEXT:NEXT:CLOSE#file
760 GCOL3,7:CIRCLE FILL (mx%*32)+16,((31-my%)*32)+16,12
770 PRINTTAB(64,10);SPC15
780 ENDPROC
790
800 DEFPROCdrawgrid
810 COLOUR128:COLOUR7:sx%=8:sy%=32:mx%=0:my%=0:VDU28,66,30,78,18,12,26
820 FOR yy%=0 TO 31
825   PRINTTAB(0,yy%);STRING$(32,CHR$224+CHR$225);
830 NEXT:GCOL3,7:CIRCLE FILL (mx%*32)+16,((31-my%)*32)+16,12
840 ENDPROC
850
860 DEF FNbyte(a%,b%)
870 p1=POINT(a%,b%):p2=POINT(a%+32,b%):p3=POINT(a%+64,b%)
                                                  :p4=POINT(a%+96,b%)
880 =(p1+(p2*4)+(p3*16)+(p4*64))
890
900 DEFPROCplot(a%,b%,c%)
910 COLOUR c%+128:COLOUR7:PRINTTAB(a%,b%);CHR$224;CHR$225;:COLOUR
                                                            128:GCOL c%
920 POINT (a%*2)+1100,((31-b%)*4)+300:POINT (a%*2)+1102,((31-b%)
                                                                *4)+300
```

```
 930 POINT a%+1140,((31-b%)*4)+110
 940 ENDPROC
 950
 960 DEFPROCmaingrid
 970 xx%=x% DIV 32:yy%=(1023-y%) DIV32
 980 IF b%=2 THEN GCOL3,7:CIRCLE FILL (mx%*32)+16,((31-my%)*32)+16
       ,12:mx%=xx%:my%=yy%:CIRCLE FILL (mx%*32)+16,((31-my%)*32)+16
 990 IF b%<>4 THEN ENDPROC                                 ,12:ENDPROC
1000 IF xx%=mx% AND yy%=my% GCOL3,7:CIRCLE FILL(mx%*32)+16,((31-my%)
                                                           *32)+16,12
1010 PROCplot((x% DIV32)*2,(1023-y%)DIV32,col%)
1020 IF xx%=mx% AND yy%=my% GCOL3,7:CIRCLE FILL(mx%*32)+16,((31-my%)
1030 ENDPROC                                               *32)+16,12
1040
1050 DEFPROCsetsize
1060 GCOL3,7:RECTANGLE FILL 0,1023,sx%*128,-sy%*32
1070 REPEAT:REPEAT:MOUSE x%,y%,b%:UNTIL b%>=2:IF b%=2 THEN 1110
1080   RECTANGLE FILL 0,1023,sx%*128,-sy%*32
1090   sx%=(x% DIV 128)+1:sy%=(1023-y%) DIV 32 +1:IF sx%>8 sx%=8
1100   RECTANGLE FILL 0,1023,sx%*128,-sy%*32
1110 UNTIL b%=2
1120 RECTANGLE FILL 0,1023,sx%*128,-sy%*32
1130 CIRCLE FILL (mx%*32)+16,((31-my%)*32)+16,12
1140 FORl%=0TO31:PRINTTAB(64-((8-sx%)*8),l%);STRING$((8-sx%)*8," ");
                                                                :NEXT
1150 IF sy%<32 :FOR l%=31 TO sy% STEP-1:PRINTTAB(0,l%);STRING$(64
                                                          ," ");:NEXT
1160 VDU28,66,30,78,18,12,26
1165 FOR ly%=1 TO sy%:FOR lx%=1 TO sx%*4
1170      PROCplot((lx%-1)*2,ly%-1,POINT((lx%*32)-16,(32-ly%)*32+16))
1171 NEXT:NEXT
1180 IF mx%>=(sx%*4) THEN mx%=(sx%*4)-1
1190 IF my%>=sy% THEN my%=sy%-1
1200 GCOL3,7:CIRCLE FILL (mx%*32)+16,((31-my%)*32)+16,12
1210 ENDPROC
1220
1230 DEF FNfilename
1240 PRINTTAB(64,10)"Name:";:COLOUR6:*FX15,0
1250 len=0:wd$="":REPEAT:REPEAT:k=GET:UNTIL k>32 AND k<128 OR k=13
1260   IF k=127 AND len=0 THEN 1300
1270   IFk=127 THENwd$=LEFT$(wd$):len-=1:VDU8,32,8:GOTO1300
1280   IF k=13 OR len=10 THEN1300
1290   len+=1:wd$+=CHR$(k):VDU k
1300 UNTIL k=13
1310 =wd$
```

```
 10 REM >SetPointer
 20 REM To set up mouse pointer
 30 REM Original program J Shipp
 40 REM New vers'n Keith McAlpine
 50 defn=0:blkptr=1:tblptr=2
           :index=3:sp=13:link=14
                    :Defpointer=&15
 60 DIM code 400
 70 FOR pass%=0 TO 2
 80   P%=code
 90   [OPT pass%
100   stmfd (sp)!,{link}
110   mov defn,#Defpointer
120   adr blkptr,parmBlock
130   adr tblptr,parmBlock+19
140   add index,blkptr,#5
150   .loop
160   strb tblptr,[index,#1]!
170   movs tblptr,tblptr,lsr #8
180   bne loop
190   swi "OS_Word"
200   ldmfd (sp)!,{pc}
210   .parmBlock
220   ]
230 NEXT
240 OSCLI("LOAD filename "+STR$
                        ~parmBlock)
  :REM change filename to own name
250 CALL code
260 MODE9:OFF
270 *POINTER
280 MOUSE COLOUR 1,parmBlock?10,
       parmBlock?11,parmBlock?12
290 MOUSE COLOUR 2,parmBlock?13,
       parmBlock?14,parmBlock?15
300 MOUSE COLOUR 3,parmBlock?16,
       parmBlock?17,parmBlock?18
310 MOUSE ON 2
320 REPEAT:REPEAT:MOUSE X,Y,B
330 UNTILB<>0:POINT X,Y:UNTIL0
```

Mouse-Art Routines

Malcolm Banthorpe

Since the introduction of the Apple Macintosh, the mouse has continued to gain in popularity with the general computer-using public. Software writers have been encouraged to make programs more user-friendly by taking advantage of the ease with which even new users are able to interact with a computer screen, particularly where no text input is required and even if the users are unfamiliar with keyboards.

The Archimedes' Welcome disc contains several demonstration programs such as the Music Editor and the Font Designer which make good use of the mouse and I'm sure that a large proportion of the commercial software to come will do likewise.

Apart from its use as a simple pointing device, the mouse can be used for drawing on the screen, as demonstrated by the Paint program on the Welcome disc. The routines described here are intended as aids to drawing with the mouse. They are not intended as finished programs but as illustrations of some procedures that I've found useful and which you can incorporate into your own software.

Some people experience an initial difficulty in freehand drawing with a mouse as, unlike drawing with a pencil on a piece of paper, the eyes cannot simultaneously monitor the drawing itself and the hand movements being used to create it. This problem is usually overcome with a little practice but it has to be admitted that, because of the way it is held, freehand drawing with a mouse is more difficult than drawing with a pencil. A graphics tablet with a pen-like stylus makes drawing easier but the purchase price is a deterrent to most of the potential users, who must therefore be content with a mouse.

The routine described here applies a variable degree of automatic smoothing to any line drawn with the mouse. To illustrate the difference this can make, first of all type in the following short sketching program and try, for instance, writing your signature on the screen. Simply press any of the three mouse buttons to make a mark on the screen.

```
MODE 12
MOUSE ON
*POINTER
REPEAT
  MOUSE X%,Y%,B%
  IF B% THEN POINT X%,Y%
UNTIL 0
```

Unless you've had a lot of practice in using a mouse in this way, you'll probably find that the lines are a bit rough.

The smoothing routine shown below works by averaging a number of coordinate pairs returned by the mouse. It does not appreciably slow down the speed at which lines can be drawn, as only one reading of the mouse coordinates is still required for every point plotted. This is achieved by taking a running average of the last, say, 100 coordinate pairs returned by the mouse. Now try drawing with this program.

```
MODE 12
MOUSE ON
*POINTER
PROCsmooth_initialise
REPEAT
  MOUSE X%,Y%,B%
  PROCsmooth(X%, Y%)
  IF B% THEN POINT X%,Y%
UNTIL 0
END

DEF PROCsmooth_initialise
pointer% = 0
sumX% = 0
sumY% = 0
smoothing% = 100
DIMX%(smoothing%),Y%(smoothing%)
ENDPROC

DEF PROCsmooth(RETURN x%,
RETURN y%)
sumX% = sumX% - X%(pointer%)+x%
sumY% = sumY% - Y%(pointer%)+y%
X%(pointer%) = x%
Y%(pointer%) = y%
x% = sumX% DIV smoothing%
y% = sumY% DIV smoothing%
pointer% = (pointer% + 1) MOD
                          smoothing%
ENDPROC
```

You should find that the lines are somewhat smoother. The variable, smoothing%, determines the number of coordinate pairs averaged for each point plotted. It has been set to 100 here but try other values to get some idea how the smoothing of the lines is affected. As the value is increased, you'll notice that the line drawn lags behind the mouse pointer and this will to some extent limit the degree of smoothing that can be used, although slowing down your hand movements will counteract this. The actual value to be used will be a matter of personal preference and will also depend on what is to be drawn, but it should be fairly easy to find one that allows you to draw smooth curves quite effortlessly. The optimum value will also depend on whether you're using ROM Basic or RAM Basic, the increased speed of the latter requiring a higher value of smoothing% to achieve the same effect.

The use of PROCsmooth need not be confined to the mouse and can in fact be used to apply smoothing to any stream of coordinates pairs, for instance from a graphics tablet or joystick. Note the use of RETURN so that whatever the variable names of the coordinates to be smoothed, the smoothed versions retain the same names.

The above programs draw lines by plotting a single point for each coordinate pair returned by the mouse. This works fine until you move the mouse too fast. Then you end up with a dotted, discontinuous line. The simplest solution is to plot a line from the current coordinates to the previous coordinates rather than just a point at the current coordinates.

Where a line thickness of greater than one pixel is required, a common technique is to use the CIRCLE FILL command to give a line whose thickness is equal to the diameter of the circles. Again, if you move the mouse too fast the result is a line of circular blobs rather than a continuous line with smooth edges.

The routine incorporated in the next demonstration program allows a continuous line of any pre-determined width to be drawn. The smoothing routine can, of course, also be incorporated by calling PROCsmooth_initialise at the head of the program before the REPEAT loop and calling PROCsmooth immediately after the MOUSE X%,Y%,B% statement.

```
MODE 12
MOUSE ON
*POINTER
PROCline_initialise
REPEAT
  MOUSE X%,Y%,B%
  IF B% PROCline(7)
  lx% = X%
  ly% = Y%
UNTIL 0
END

DEF PROCline_initialise
  x% = 0
  y% = 0
  lx% = 0
  ly% = 0
  width% = 8
ENDPROC

DEF PROCline(colour)
GCOL 0, colour
dX% = X% - lx%
dY% = Y% - ly%
L = width% / (SQR(dX% * dX% +
              dY% * dY%) + 1E-6)
x% = dY% * L
y% = dX% * L
MOVE lx% + x%, ly% - y%
MOVE lx% - x%, ly% + y%
PLOT 85, X% + x%, Y% - y%
PLOT 85, X% - x%, Y% + y%
ENDPROC
```

The variable, width%, which is set in PROCline_initialise determines the width of the line drawn. The parameter, colour, determines the logical colour in which the line will be drawn.

The addition of three more lines to the procedure, PROCline, will allow lines of any width to be drawn, which are outlined in a different colour:

```
GCOL 0, 2
LINE x1%, y1%, X% + x%, Y% - y%
LINE x2%, y2%, X% - x%, Y% + y%
```

These lines should be inserted immediately preceding the ENDPROC in the definition of PROCline. As shown, the program will draw white lines outlined in green. A second parameter could be added to the PROCline definition to set the outline colour.

Many microcomputer painting programs offer an 'airbrush' or 'spray' option in which the effect of an airbrush is simulated by plotting a series of points, the exact position of each deviating from the current cursor position by a controlled random distance. If the points are small enough, such as in Archimedes mode 12, the result is a reasonable representation of an airbrush effect. The more dense the distribution of points on the screen, the denser the colour appears.

However, with the range of definable colours and the processing speed available on the Archimedes, it's possible to go one step better than this with the simulation and make the density of colour at any particular point on the screen increase as more 'drops of paint' fall on it. In this example I've used mode 12 with the logical colours redefined to give 15 levels of blue. Other possibilities you might like to experiment with are to have seven levels each of two different hues or to use a 256 colour mode to increase the colour range (although in these cases the processing becomes more complicated to allow the mixing of two or more colours) .

```
MODE 12
FOR I% = 0 TO 15
  C% = (15 - I% ) * 16
  COLOUR I%, C%, C%, 255
NEXT
radius% = 32
radius2% = radius% * radius%
MOUSE ON
*POINTER
REPEAT
 MOUSE X%,Y%,B%
 IF B% THEN
  REPEAT
   x% = radius - RND(radius<<1)
   y% = radius - RND(radius<<1)
  UNTIL x% * x% + y% * y% <
                          radius2%
  X% += x%
  Y% += y%
  IF B% = 1 THEN
   C% = POINT(X%, Y%) - 1
   IF C% < 0 C% = 0
  ELSE
   C% = POINT(X%, Y%) + 1
   IF C% = 16 C% = 15
  ENDIF
  GCOL 0, C%
  POINT X%, Y%
 ENDIF
UNTIL0
```

The program gives a white background onto which blue 'paint' can be sprayed by pressing the left-hand mouse button. Pressing the right-hand button has the effect of spraying white 'paint' so that parts of the image already drawn in blue can be progressively lightened and finally erased. The variable, radius%, determine the radius of the paint spray. PROCsmooth can also be incorporated but there is probably little to be gained by doing so.

The program works as follows: The logical colours are redefined so that 0 is white and then as the number is increased towards 15, the red and green components are reduced so that the colours range from very pale blue for colour 1 to fully saturated blue for colour 15.

When the left-hand mouse button is pressed, for each pair of mouse coordinates, a displaced coordinate pair is generated which lies within the pre-determined spray radius. The colour of the screen at that point is then read and if it is less than 15 it is incremented and the new colour plotted. If the right-hand button is pressed and the colour is greater than 0 then it is decremented. As a further variation, try replacing POINT X%, Y% with CIRCLE FILL X%, Y%, 4 . A

AMaze – The Maze Generator Game

AMaze is a random maze generator program for the **Archimedes** computer, it generates mazes which always have no more or less than one negotiable route through them. Two games are supplied giving alternative methods of tracing a path through the mazes. If you negotiate a maze successfully, you will be presented with another harder one.

The program is written in BASIC V. The mazes are drawn as they are created and this process is quite fascinating to watch.

Price £5 inclusive, £4 if ordered before end March '88

Order from : W.R.Hindle, 18 Hoober Rd, Sheffield S11 9SF

Order form

Please supply 1 copy of AMaze to the address below. I enclose cheque/Postal order for £4.

Name ------------------

Address ------------------

PINEAPPLE SOFTWARE

DIAGRAM II for the ARCHIMEDES

Diagram II represents a major breakthrough in the techniques used for drawing software on the BBC micro. It works on a completely different principle to other drawing software by storing the drawing information straight to disc as coded 8*8 pixel blocks.

This technique has three major advantages. First, the size of the diagram is only limited by the amount of space on disc. Second, there is no limit to the amount of information which can be stored on a given diagram as the information is stored on disc and not in computer memory. Third, the disc storage technique allows the smooth scrolling of the screen over the whole surface of the diagram. Up to about 120 Mode 0 screens on an 800k Archimedes disc.

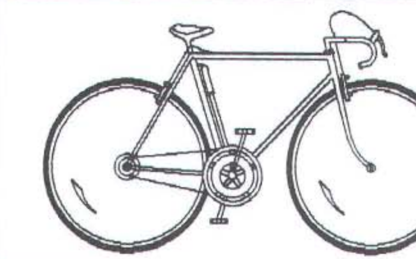
For people not familiar with Diagram, the basic operation of the program is to first display any part of the diagram, either by quoting an index name or screen number. At this point you are free to scroll the diagram around the screen using either cursor keys, or Archimedes mouse. You may stop scrolling at any point and enter the edit mode which allows you to modify or add new information to the diagram. When you have finished making alterations, the new information is stored to disc and you are free to scroll elsewhere on the diagram stopping to edit at any time.

The edit features are now very comprehensive. Text may be typed straight from the keyboard, and if other fonts are stored as user defined characters then these fonts may be printed using the normal keyboard keys.

The print routines are now completely 'scaleable' with both the horizontal and vertical scales being totally variable in 1% steps between a size that will give 18 mode 0 screens on an A4 sheet and a size that would print just one pixel on a sheet! The routines are adaptable to most types of dot matrix printer and full advantage may be made of wide carriage printers. The printouts may be rotated through 90 deg. if required.

The new editing features make DIAGRAM II suitable for all types of serious drawing application including scale drawings, flow charts, architectural, family trees, and many other subjects as well as circuit and schematic diagrams. As an example of what Diagram II can do, this complete advertisement has been produced with the package.

1. Provides up to 880 User Definable Characters stored in memory. Rapid horizontal and vertical line drawing routine with automatic joins for circuit diagrams.
2. Full rubber band line drawing and circle drawing modes.
3. Takes advantage of the Extended graphics facilities of the Archimedes to provide drawing of arcs, sectors, chords, parallelograms, ellipses and flood filling of areas.
4. Pixel cursor drawing and deleting mode allows very fine detail to be added.
5. Defined areas of screen may be moved, copied, deleted or saved to disc effectively allowing unlimited numbers of icons up to a full screen size each.
6. On-screen indication of cursor position shows either the cursor position on the overall diagram or the distance from a preset point.
7. Keyboard keys may be predefined to print User Defined Characters, enabling new character sets to be stored as user defined characters and then printed straight to the screen using the keyboard keys.
8. Wordprocessor files may be loaded to the screen and automatically formatted into any shape screen area.
9. Index names may be set up to point to given areas of a large diagram, to enable rapid access to a given point on a diagram.
10. Fully compatible with the Archimedes Mouse.
11. All Diagram 'Utilities' are included, enabling screen ident numbers and borders to be added, and any area of diagram to be moved or copied. The whole diagram may be displayed in reduced scale (either 4*4 or 8*8 format) on a single screen, and the size of the overall diagram may be increased or decreased.
12. Completely scaleable print routines allow any area of the diagram to be printed either horizontally or through 90 deg. in scales that may be varied in 1% steps allowing up to 18 mode 0 screens to be printed on an A4 sheet (still with readable text).
13. Complete with 40 page easy to understand handbook.



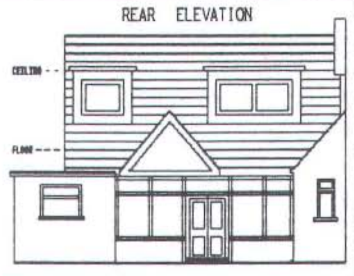


Diagram II runs on the Archimedes 6502 emulator but at about 4 times the speed of the BBC version. It is supplied on 3.5" 800k disc.

DIAGRAM II – £55.00 + vat

P & P free (except overseas)

N.B. Upgrades are available to existing Diagram owners. Please contact Pineapple if you have not received an upgrade letter.

VISA

39 Brownlea Gardens, Seven Kings, Ilford, Essex IG3 9NL Tel 01-599 1476

Access

The WIMP Environment – Part 3

Adrian Look

As promised in part 2 of this series (Archive issue 3), I shall show you how to 'write' into the windows. You guessed it! The Arthur Window Manager (AWM) has a special system for updating and writing into windows. Indeed it would be foolish for the programmer to attempt to write directly into the windows. This would cause all sorts of problems!

In part 2, I described the 'Wimp_Poll' subroutine. You may remember one of the conditions serviced by this routine was a 'redraw window request'. I shall now explain how to use this to 'legally' update and write into the windows you have defined.

Once the windows have been set up the AWM monitors them. If the user changes the windows in any way, the AWM will have to redraw the parts of the system which have been affected. The AWM can sometimes do all the redrawing itself but since it does not always know exactly what is contained in the window, if the visible part of the window is changed in any way, the AWM will create a 'redraw window request' so that the application can draw the relevant information.

The 'redraw window request' is recognised by a reason code (as described in part 2) of value 1 while in the 'Wimp_Poll' loop. On receipt of this reason code, the program should call a subroutine which redraws the part of the window which AWM wants to be redrawn.

The handle of the window to be redrawn is stored at the beginning of the 'Wimp_Poll' block. Using this, the update routine should call the 'Wimp_RedrawWindow' subroutine

'Wimp_RedrawWindow' at &400C8

This routine will redraw the window itself, i.e. the border, and any other parts of the visible area it can. It then splits the parts it cannot redraw into rectangles and makes a list of them. The routine exits via the 'Wimp_GetRectangle' routine.

'Wimp_GetRectangle' at &400CA

This routine calls the next rectangle to be redrawn from the list. It then puts the following information about the rectangle into a block (which the programmer defines before entering into the routine). On exiting this routine returns a flag which indicates whether there are any more rectangles to be redrawn. The information in the block is as below:

```
 0 - window handle
 4 - work area min x (x0)
 8 -           min y (y0)
12 -           max x (x1)
16 -           max y (y1)
20 - x scroll position (scx)
24 - y scroll position (scy)
28 - current graphics window min x
32 -                         min y
36 -                         max x
40 -                         max y
```

Having managed to get this information, we can now update the window. Note that a VDU5 has been performed and any text should conform to this when updating the window. Here is an example of how to service a 'redraw window request':

```
DIM block% &100
REPEAT
SYS "Wimp_Poll",,block% TO reason
CASE reason OF
:
WHEN 1:PROCredraw_window(!block%)
:
ENDCASE
UNTIL FALSE
END
:
```

```
DEFPROCredraw_window(handle)
!block%=handle
SYS "Wimp_RedrawWindow",,block%
                              TO more
CASE handle OF
:
WHEN window : PROCdraw_window
:
ENDCASE
:
DEFPROCdraw_window
WHILE more
PROCinfo
:
REM perform drawing required
:
SYS "Wimp_GetRectangle",,block%
                              TO more
ENDWHILE
ENDPROC
:
DEFPROCinfo
x0=block%!4 : REM work area
                          coordinates
y0=block%!8
x1=block%!12
y1=block%!16

scx=block%!20 :REM scroll posit'n
scy=block%!24

gx0=block%!28 :REM graphics
               window co-ordinates
gy0=block%!32
gx1=block%!36
gy1=block%!40
ENDPROC
```

Of course it is not only the WIMP environment that may change the contents of the windows. The application may update the window of its own accord. In which case the AWM will not be able to issue a 'redraw window request' as it does not know about the update. For this purpose the subroutines below were devised.

'Wimp_UpdateWindow' at &400C9

This routine allows the user to completely redraw the visible area of the window. The routine requires a block with the window's handle and its work area extent co-ordinates. The routine then compiles all the visible rectangles of that window and makes a list of them. The routine then proceeds the same as for the 'Wimp_RedrawWindow'. The entry block is as below:

```
 0 - window handle
 4 - work area min x (x0)
 8 -           min y (y0)
12 -           max x (x1)
16 -           max y (y1)
```

An example subroutine:

```
DIM block% &100
:
DEFPROCupdate_window(handle)
SYS "Wimp_GetWindowState",,block%
SYS "Wimp_UpdateWindow",,block%
CASE handle OF
:
WHEN window : PROCdraw_window
:
ENDCASE
ENDPROC
```

'Wimp_ForceRedraw' at &400D1

This routine forces the AWM to issue a 'redraw window request' for a specific area. This is called as below:

```
SYS "Wimp_ForceRedraw",handle,rx0
                         ,ry0,rx1,ry1
```

where (rx0,ry0) and (rx1,ry1) are the co-ordinates of the area to be redrawn. Note that a handle of –1 causes the whole screen to be redrawn, as though the AWM had issued the request. i.e. through the 'Wimp_Poll' routine.

So when updating a window, the programmer can either force a redraw or update the whole window using either of the above routines. The choice of methods is entirely up to you.

To finish, I have decided to include a complete example program of what we have done so far so that you may see how it all links together.

Next month I shall describe how to use the AWM's in-built text input routines.

```
 10 REM >$.Wimp.Drawing
 30 REM************************
 40 REM*Drawing in WIMP Windows*
 50 REM* By Adrian Philip Look *
 60 REM*  30th December 1987   *
 70 REM************************
 90 ON ERROR OSCLI("FX4,0")
          :MODE12:PRINT REPORT$;
             " at line ";ERL:END
110 *PRINT $.Wimp.WimpFont
120 MODE12
130 FOR colour=8 TO 15
140   COLOURcolour,0,0,255
150 NEXT colour
160 PROCvariables
170 version=FNwimp_initialise
190 flags=FNflags(TRUE,TRUE,TRUE
          ,TRUE,TRUE,TRUE,FALSE)
200 title_type=FNtitle_type(TRUE,
            TRUE,TRUE,TRUE,TRUE)
220 wally%=FNcreate_window(50,50,
         500,500,1000,1000,flags,
       "Window 1",title_type,top)
230 hello%=FNcreate_window(400,
       400,850,850,500,500,flags,
       "Window 2",title_type,top)
240 anext%=FNcreate_window(800,
    200,1000,700,1000,1000,flags,
       "Window 3",title_type,top)
250 which%=FNcreate_window(50,
     700,400,800,1000,1000,flags,
       "Window 4",title_type,top)
270 PROCopen_window(wally%,TRUE)
280 PROCopen_window(hello%,TRUE)
290 PROCopen_window(anext%,TRUE)
300 PROCopen_window(which%,TRUE)
320 SYS "Wimp_ForceRedraw",-1,0,
                    0,1280,1024
330 *POINTER
340 REPEAT
350   SYS "Wimp_Poll",mask,
                 block% TO reason
360   handle=!block%
370   PROCwimp_poll(reason,
                         handle)
380 UNTIL FALSE
390 END
400
410 DEFPROCvariables
420 DIM block% &100
430 title_foreground=1
440 title_background=6
450 work_area_foreground=1
460 work_area_background=7
470 scroll_bar_inner=1
480 scroll_bar_outer=6
490 top=-1
500 mask=0
510 amountx=100
520 amounty=100
530 ENDPROC
540
550 DEFPROCwimp_poll(reason,
                         handle)
560 *FX19
570 CASE reason OF
580   WHEN 1:PROCredraw(!block%)
590   WHEN 2:PROCopen_window
                 (handle,FALSE)
600   WHEN 3:PROCclose_window
                       (handle)
610   WHEN 1:PROCscroll_
                    acknowledge
620 ENDCASE
630 ENDPROC
640
650 DEFFNwimp_initialise
660 LOCAL version%
670 SYS "Wimp_Initialise",0 TO
                       version%
680 =version%/100
690
700 DEFFNcreate_window(x0,y0,x1,
   y1,sizex,sizey,flags,title$
          ,title_type,pos)
710 block%!0=x0
720 block%!4=y0
730 block%!8=x1
740 block%!12=y1
760 block%!16=0
770 block%!20=0
790 block%!24=pos
810 block%!28=flags
830 block%?32=title_foreground
```

```
 840 block%?33=title_background
 850 block%?34=work_area_
                        foreground
 860 block%?35=work_area_
                        background
 870 block%?36=scroll_bar_outer
 880 block%?37=scroll_bar_inner
 890 block%?38=0
 910 block%!40=0
 920 block%!44=-sizey
 930 block%!48=sizex
 940 block%!52=0
 960 block%!56=title_type
 980 $(block%+72)=LEFT$
                      (title$,11)
1000 block%!84=0
1020 SYS "Wimp_CreateWindow",0,
                 block% TO handle%
1030 =handle%
1040
1050 DEFFNflags(tbar,move,vscb,
            hscb,boqb,scrl,wimp)
1060 LOCAL flag%
1070 flag%=0
1080 IF tbar THEN flag%=flag%OR&01
1090 IF move THEN flag%=flag%OR&02
1100 IF vscb THEN flag%=flag%OR&04
1110 IF hscb THEN flag%=flagOR&08
1120 IF wimp THEN flag%=flag%OR&10
1130 IF NOT boqb THEN flag%=flag%
                             OR&80
1140 IF scrl THEN flag%=flag%OR
                             &100
1150 =flag%
1160
1170 DEFFNtitle_type(hasb,text,
                 txhc,txvc,back)
1180 LOCAL flag%
1190 IF hasb THEN flag%=flag%OR&04
1200 IF text THEN flag%=flag%OR&01
1210 IF txhc THEN flag%=flag%OR&08
1220 IF txvc THEN flag%=flag%OR&10
1230 IF back THEN flag%=flag%OR&20
1240 =flag%
1250
1260 DEFPROCopen_window(handle,
                             new)
1270 IF new THEN
1280   block%!0=handle
1290   SYS "Wimp_GetWindowInfo"
                         ,,block%
1300   block%!28=top
1310 ENDIF
1320 SYS"Wimp_OpenWindow",,block%
1330 ENDPROC
1340
1350 DEFPROCclose_window(handle)
1360 !block%=handle
1370 SYS"Wimp_CloseWindow",,block%
1380 ENDPROC
1390
1400 DEFPROCdelete_window(handle)
1410 !block%=handle
1420 SYS "Wimp_DeleteWindow",,
                           block%
1430 ENDPROC
1440
1450 DEFPROCscroll_acknowledge
1460 scx=block%!20
1470 scy=block%!24
1490 scx+=(block%!32)*amountx
1500 scy+=(block%!36)*amounty
1520 block%!20=scx
1530 block%!24=scy
1540 SYS"Wimp_OpenWindow",,block%
1550 ENDPROC
1560
1570 DEFPROCredraw(handle)
1580 !block%=handle
1590 SYS "Wimp_RedrawWindow",,
                  block% TO more
1600 PROCwhich_window
1610 ENDPROC
1620
1630 DEFPROCwhich_window
1640 CASE handle OF
1650   WHEN hello%
              :PROCdraw_example1
1660   WHEN wally%
              :PROCdraw_example2
1670   WHEN which%
              :PROCdraw_example3
1680   WHEN anext%
              :PROCdraw_example4
1690 ENDCASE
1700 ENDPROC
1710
1720 DEFPROCdraw_example1
1730 WHILE more
```

```
1740   PROCinfo
1760   MOVE bx+175,by-175
           :PLOT &95,bx+20,by-175
1770   MOVE bx+32,by-143
         :PRINT"  Using the WIMP"
1780   MOVE bx+32,by-175
           :PRINT"    by Archive"
1800   SYS "Wimp_GetRectangle",,
                     block% TO more
1810 ENDWHILE
1820 ENDPROC
1830
1840 DEFPROCdraw_example2
1850 WHILE more
1860   PROCinfo
1880   gx0=(gx0-bx)DIV32 *32 +bx
1890   gy1=(gy1-by)DIV32 *32 +by
1900   FOR y=gy1TOgy0-64 STEP -32
1910     FOR x=gx0-64TOgx1 STEP32
1920       GCOL(x/32+y/32)MOD6+1
1930       MOVE x+16,y+16
               :PLOT &9d,x+2,y+16
1940   NEXTx:NEXTy
1960   SYS "Wimp_GetRectangle",,
                     block% TO more
1970 ENDWHILE
1980 ENDPROC
1990
2000 DEFPROCdraw_example3
2010 WHILE more
2020   PROCinfo
2040   MOVE bx,by
2050   FOR temp=0 TO 200
2060     GCOL 0,temp MOD 6+1
2070     PRINT"Hello and Welcome
                      to Archive";
2080     IF(tempMOD5)=0 THEN PRINT
2090   NEXT temp
2110   SYS "Wimp_GetRectangle",,
                     block% TO more
2120 ENDWHILE
2130 ENDPROC
2140
2150 DEFPROCdraw_example4
2160 WHILE more
2170   PROCinfo
2190   GCOL 0,RND(5)
2200   PLOT 141,gx0+1,gy0+1
2220   SYS "Wimp_GetRectangle",,
                     block% TO more
2230 ENDWHILE
2240 ENDPROC
2260 DEFPROCinfo
2270 x0=block%!4
2280 y0=block%!8
2290 x1=block%!12
2300 y1=block%!16
2320 scx=block%!20
2330 scy=block%!24
2350 gx0=block%!28
2360 gy0=block%!32
2370 gx1=block%!36
2380 gy1=block%!40
2400 bx=x0-scx   : REM all drawing
       should be relative to bx,by
2410 by=y1-scy
2420 ENDPROC
2430
2440 DEFPROCupdate_window(handle)
2450 !block%=handle
2460 SYS "Wimp_GetWindowState",,
                             block%
2470 SYS "Wimp_UpdateWindow",,
2480 PROCwhich_window        block%
2490 ENDPROC
```

Matters Arising...

I wish to apologise for some errors that occurred in the first article of this series and I hope that this has not caused too much confusion. The corrections are as follows:

(1) Directly after every SYS "Wimp_" command there should be a ",0".

For example:

SYS "Wimp_Initialise" TO version% *should read* SYS "Wimp_Initialise",0 TO version%

SYS "Wimp_CloseWindow", handle_block% *should read* SYS "Wimp_CloseWindow",0, handle_block%

(2) In the procedure DEF PROCopen_window, page 19, the line: open_block%!4=y1 *should read* open_block%!4=x0.

Screen Dump for Epson Printers

Relocatable Module Version

Gerald Fitton

Having given us a BASIC function for an Epson screen-dump and, last month, a relocatable machine code version, Gerald now turns his attention to the idea of putting the code into a module, providing us with the command: *Minidump. But first a bit of background...

In the days of long ago, whenever I needed a storage location for a variable, say x%, then I would have to decide where, in the memory of the machine, this value would be stored. Nowadays, the BASIC command LET x%=6 relieves the programmer of this chore. As a general rule it is better to let the language, environment or operating system look after its own memory management.

The Model B and the Master have a 6502 central processor with an instruction set that uses absolute addressing: this implies that a machine code program will run only if it is loaded into a particular address. For those machines the memory management of machine coded programs has to be the responsibility of the programmer. The 8086 series of central processors, used in IBM PCs and their clones, suffer from the same limitations.

The Archimedes, however, uses an instruction set which, for the most part, uses addresses which are expressed as the difference, positive or negative, from the address of the current instruction. Such addressing is called Relative Addressing and the difference is called the offset. The exceptions are SoftWare Interrupt (SWI) calls which are routed to an address fixed by the operating system. It is the instruction set of the CPU which makes it possible to write code which will run (almost) anywhere in memory. For future compatibility, and because it does it better, it is desirable to leave memory management to the excellent facilities provided by Acorn.

Relocatable Code or Relocatable Module

In BASIC, a space can be reserved for machine code with DIM code% as we saw previously. We do not need to know where code% is situated: we just use the BASIC command CALL code% and the system finds the right place. When we have finished with that piece of code we can load another utility into the same address space and CALL it to do a different job. If memory is short, being able to use the same address space in this way is useful. The operating system, Arthur, provides a Relocatable Module Area (RMA) into which, disc files which have been classified as modules (with *SETTYPE &FFA), are automatically loaded; Archimedes integrates them into the rest of the system. This relieves the programmer of the associated memory management at the cost of complying with the RM Manager's rules. Modules can be languages (e.g. BASIC), filing systems (e.g. ADFS), new SWI calls (e.g. Window Manager) or utilities called with a *<command>. A *<command> utility which is part of a relocatable module can be called from within any environment and not only from BASIC. We shall make the screen dump of this series of articles a new utility which is executed by the command *MiniDump with three optional parameters.

The MiniDump Module Header

Any piece of relocatable machine code can be turned into a relocatable module by adding a suitable piece of code, called a header, in front of it. The function of the header is to enable the OS to find the code, recognise whether it is a language, new SWI call etc, provide help and integrate it into the system generally. For this article I am providing the source code of a fairly simple header, which, with a few modifications, could be placed in front of almost any piece of relocatable machine code so that the code could then be called with a *<command> having a number of parameters.

The Header Sections

The source code for the header is written as a self-contained function stored in a disc file called RHsourceFN. The function is DEFined at line 50010 as FNassemble_MiniDumpAP (start%) having the parameter start%. Default values for the three *MiniDump parameters are assigned at lines 50160 to 50190. The minimum number of parameters is set to zero at line 50340. Since the function is self-contained, the registers used for margin etc can be different from the relocatable code function (50240 to 50260).

The module header consists of three sections. The first section includes eleven four byte words which define the function of the module and provide pointers to the keywords. The second section consists of a block for each keyword. We are using only one keyword, *MiniDump. The third section does the real work. The code in it reads and validates the values of the parameters and stores them in the registers R0 to R2 ready for use by the (already tested) screen dump code which is assembled from the source code (self-contained) function RCsourceFN.

Section 1

Of the eleven words at the beginning of the module (50410 to 50510), the first four words and the last four relate to the use of a module as a language, filing system or as a means of extending the number of SWI calls. We only need the middle three words (lines 50450 to 50470). These contain the addresses of the module title, module help message and the table of keywords. All these addresses are relative to the start of the module. The "moduletitle" which appears at line 50520 is a BASIC variable declared as LOCAL at line 50060. This BASIC variable contains the absolute address of the code which starts at line 50520. The start of the code is stored in the BASIC variable start% passed as a parameter to the function (50390). The subtraction at line 50450 ensures that the relative address is stored in the module code. The string at 50530 is the title of the module which is used by Archimedes when *MODULES is called and it is used by *RMKILL <title>. The module help string (50570) appears when *HELP Modules is used.

Section 2

The keyword "MiniDump" assigned at line 20610 will be displayed only by the OS call, *HELP Commands. The keyword help strings (50740 to 50780) are displayed by *HELP MiniDump. The block of code from 50610 to 50690 must be repeated for each keyword. We have only one. When *MiniDump is called, Archimedes checks whether the number of parameters is between the values declared in the word of 50650 and 50660. The bad syntax error message (50710) is printed if too many or too few parameters are included in the command. The offset from the start to the code at readparameters (51040) is calculated (50640).

Section 3

The code from 51040 to 51390 reads and validates the values of the parameters which follow *MiniDump. If no parameters are typed in after *MiniDump, the default values are those allocated at lines 51060 to 51080. If the parameters are out of range then the overflow flag (V) must be set and R0 should point to a suitable error message. These error messages are stored at lines 50810 to 51000. After such an error the code at **fail** (51010) which returns control to the previous environment (usually

BASIC) is executed. Line 51020 sets the V flag and 51030 returns from the module.

I shall take as an example "*MiniDump 11 2" which gives a margin of 11 tenths of an inch, a threshold colour of 2 and the default value of 7 for printmode. On entry to the module, R0 contains the address of the string following *MiniDump, i.e. the "11 2"; R1 contains the number of parameters, in this case two. Lines 51090 and 51100 re-allocate these to R1 and R3 respectively ready for the call "OS_ReadUnsigned" of 51140. In preparation for this call, the address of the **badparameters** error message is stored in R0 by line 51110. If the call fails, the message "Bad Parameters" will be displayed. I have been unable to cause "OS_ReadUnsigned" to fail so I have not tested this piece of code.

At 51120, the number of parameters (two in our example) is compared with 0. If there are 0 parameters then the program branches to the end, **storeparameters** at line 51400. The OS call returns into R2 the value of the next parameter in the string, in our case 11. After setting up the error message at line 51160, this value of margin is compared with 79 at 51170. If the margin is greater than 79 then a branch to **fail** is executed and the **badmargin** string is printed. Similarly, the value of **threshold** is extracted from the string following *MiniDump (51220), compared with 255 at 51250 and then, at 51280 the number of parameters in the string (two) is compared (successfully) with 2 and a branch to **storeparameters** is executed at 51290. The effect of this code is to place the values 11, 2 and 7 into the registers called margin, threshold and printmode (i.e. R4, R5 and R6).

The short piece of code at 51400 to 51430 moves these values into R0, R1 and R2 ready for use by the relocatable screen dump code. Line 51440 restores the values of R3 to R12 pushed onto the stack at line 51030 and pulls the link back into R14. If you wish to test this piece of code without doing a screen dump every time you should change line 51440 to LDMFD (sp)!,{R3-R12,PC}. With this change you can check the format of your help and error messages.

At line 51540 the function returns the address of the end of the code. This is the address to which you have to *LOAD your relocatable machine code before saving the composite module.

The Programs

There are three programs. The first, RHsourceFN, is the source program, written as a self-contained function. The linker, RMlinker, is very similar to RClinker and you may use that as a starting point. Type in the two extra lines 1020 and 1070; then modify 360 and 1140. The effect of the two lines 1070 and 1080 is to assemble the two blocks of code consecutively using pointer% as a variable holding the values of the parameters start% and end%. At 1130 the composite is *SAVEd under the name given in line 360.

The third program, MainProgRM, is an example in BASIC showing how the module is loaded and called. If you get a "No room" message, you may have to *RMTIDY to make more room in the RMA (Relocatable Module Area).

Over to You

Perhaps your Epson has a worn ribbon! In that case, double or treble printing each line will help. Referring to the MiniDumpFN of 26th October 1987, you can produce treble printing with 1/216" line feeds by adding the following lines:-

```
50512 double%=passes% :REM No. of
       passes = 1, 2 or 3
50514 REPEAT
50672 VDU 27,74,1,13:REM LF=1/216
50674 double%-=1
50676 UNTIL double%=0
50680 VDU 27,74,24-passes%,13
       :REM This was VDU 13,10
```

Now convert that to machine code! [Hint: Put "bottom" on the stack.]

If you have an early Epson with graphics print mode 0 or 1 this will give only 60 or 120 dots per horizontal inch causing elliptical distortion. The new LQ series do not support modes 5 or 7. However, within the limitations of the printer and armed with this series of programs you should be able to achieve results. In any case, you should now be able to add a relocatable module header to any relocatable code having your own *<command>.

Happy Computing!

```
 100 REM > MainProgRM
 110 REM BASIC version 3 :5/12/87
 120 REM © ABACUS TRAINING
 160 :
 170 ON ERROR:CLOSE#0:VDU4,26,12
        :OSCLI("FX3"):REPORT:PRINT
              " at line ";ERL:STOP
 190 MODE 1
 220 margin% = 18:REM Left margin
               in tenths of an inch
 230 threshold%= 0 :REM Logical
    colour threshold for printing
 240 printmode%= 5 :REM Printmode
                    must be 5 or 7
 250 :
1000 REM The relocatable module
               must be loaded with
                 *RMLOAD MiniDumpRM
1010 OSCLI("RMLOAD $.DumpProgs.
   MiniDumpRM"):REM If "no room",
                    QUIT and retry
1020 REM Print VDU coded picture
1030 *PRINT $.DumpProgs.picture1
1040 :
1050 REM *MiniDump looks like an
        extension of the operating
                  system commands.
1060 REM It can be called by
                   OSCLI(string$).
1070 REM OSCLI("MiniDump") Would
       use the default parameters.
1080 OSCLI("MiniDump "+STR$
   (margin%)+" "+STR$(threshold%)
         +" "+STR$(printmode%))
1090 REM Parameters are optional
      with default values 18 0 7.
1100 REM *MiniDump [<margin>
     [<threshold> [<printmode>]]]
                   could be used.
1110 END
```

```
 100 REM > RMlinker : Assembles
  & links MiniDumpRM object code.
 110 REM Version 3.00 : 5/12/87
 120 REM ©ABACUS TRAINING
 130 :
 150 REM Assembles & links
    object code which will run as
             relocatable module.
 160 REM The assembled code is a
          mini screen dump with 3
                     parameters.
 180 REM <margin> Left margin.
 190 REM <threshold> Pixelcolour
     >threshold then print a dot.
 200 REM <printmode> Printer
          graphics mode (5 or 7).
 220 :
 230 REM Define effect of error.
 240 ON ERROR VDU 26,12:REPORT
      :PRINT " at line ";ERL:STOP
 260 MODE 3
 290 REM Global variables.
 300 DIM objectcodestart% &600
    :REM Room for the object code
 310 objectcodeend% =0 :REM
    Address of end of object code
 320 pointer% =0 :REM Address of
      start of each block of code
 330 :
 350 pathname$="$.DumpProgs."
 360 filename$="MiniDumpRM"
    :REM File name of object code
 370 :
1000 REM Core section.
1010 REM Load FNs containing the
       two blocks of source code
1020 LIBRARY "$.DumpProgs.
                     RHsourceFN"
   :REM Relocatable module header
1030 LIBRARY "$.DumpProgs.
 RCsourceFN":REM Relocatable code
 1050 pointer%=objectcodestart%
       :REM Initialise the pointer
 1070 pointer%=FNassemble_
             MiniDumpAP(pointer%)
        :REM Reset to end of block
 1080 pointer%=FNassemble_
             MiniDumpAC(pointer%)
        :REM Reset to end of block
 1100 objectcodeend%=pointer%
 1130 OSCLI("SAVE "+pathname$
                  +filename$+" "+
         STR$~(objectcodestart%)+
        " "+STR$~(objectcodeend%))
            :REM Save object code
 1140 OSCLI("SETTYPE "+pathname$+
               filename$+" &FFA")
          :REM Set type as Module.
 1150 OSCLI("STAMP "+pathname$
                      +filename$)
 1180 END     :REM Date stamp file.


50000 REM > RHsourceFN :Assembler
     FN to generate module header
50010 DEF FNassemble_
               MiniDumpAP(start%)
50020 REM FN Version 3.0: 5/12/87
50030 REM ©ABACUS TRAINING
50040 :
50050 LOCAL margin,threshold,
          printmode,sp,pass%,end%
50060 LOCAL modulehelp,keywords,
       moduletitle,readparameters
50070 LOCAL minparameters,
                    maxparameters
50080 LOCAL badsyntax,keywordhelp
50090 LOCAL margindefault,
 thresholddefault,printmodedefault
50100 LOCAL fail,badparameters,
badmargin,badthreshold,badprintmode
50110 LOCAL ERROR
50120 ON ERROR OSCLI("FX 3,0")
          :VDU 26,12:REPORT:PRINT
             " at line ";ERL:STOP
50160 REM Choose default values
        of the three parameters.
50170 margindefault     = 18
50180 thresholddefault =  0
50190 printmodedefault =  7
50200 :
50230 REM Registers used for
                      parameters.
50240 margin      =  4
50250 threshold   =  5
50260 printmode   =  6
50280 :
50290 REM Use the BASIC stack.
50300 sp = 13   :REM Stack pointer
50320 :
50330 REM Set minimum and maximum
              number of parameters.
50340 minparameters = 0
50350 maxparameters = 3
50370 :
50380 FOR pass% = 0 TO 3 STEP 3
50390   P%=start%
50400   [OPT pass%
50410   EQUD 0
50420   EQUD 0
50430   EQUD 0
50440   EQUD 0
50450   EQUD (moduletitle-start%)
50460   EQUD (modulehelp-start%)
50470   EQUD (keywords-start%)
50480   EQUD 0
50490   EQUD 0
50500   EQUD 0
50510   EQUD 0
50520   .moduletitle
50530   EQUS "MiniDumpRM"
50540   EQUB 0
50550   ALIGN
50560   .modulehelp
50570   EQUS "ScreenDump"+CHR$(9)
     +"3.0 (5/12/87)ABACUS TRAINING"
50580   EQUB 0
50590   ALIGN
50600   .keywords
50610   EQUS "MiniDump"
50620   EQUD 0
50630   ALIGN
50640   EQUD (readparameters-
                           start%)
50650   EQUW minparameters
50660   EQUW maxparameters
50670   EQUD (badsyntax-start%)
50680   EQUD (keywordhelp-start%)
50690   EQUD 0
50700   .badsyntax
50710   EQUS "Syntax:"+CHR$(9)+
  "*MiniDump [<margin>[<threshold>
       [<printmode>]]]"+CHR$(&0D)
50720   EQUB 0
50730   ALIGN
50740   .keywordhelp
50750   EQUS "*MiniDump"+CHR$(9)+
          "[<margin> [<threshold>
       [<printmode>]]]"+CHR$(&0D)
50760   EQUS CHR$(&09)+CHR$(9)+
          "<margin> between 0 and
                  79"+CHR$(&0D)
50770   EQUS CHR$(9)+CHR$(9)+
       "<threshold> between 0 and
                 255"+CHR$(&0D)
50780   EQUS CHR$(9)+CHR$(9)+
        "<printmode> must be 5 or
                   7"+CHR$(&0D)
50790   EQUB 0
50800   ALIGN
50810   .badparameters
50820   EQUD &00800100
50830   EQUS "Bad Parameters"
50840   EQUB 0
50850   ALIGN
50860   .badmargin
50870   EQUD &00800100
50880   EQUS "<margin> must be
              between 0 and 79"
50890   EQUB 0
50900   ALIGN
50910   .badthreshold
50920   EQUD &00800100
50930   EQUS "<threshold> must be
             between 0 and 255"
50940   EQUB 0
50950   ALIGN
50960   .badprintmode
50970   EQUD &00800100
50980   EQUS "<printmode> must be
                 either 5 or 7"
50990   EQUB 0
51000   ALIGN
51010   .fail
51020   CMP R15,#&80000000
51030   LDMFD (sp)!,{R3-R12,PC}
51040   .readparameters
51050   STMFD (sp)!,{R3-R12,R14}
51060   MOV margin,#margindefault
51070   MOV threshold,
                #thresholddefault
51080   MOV printmode,
                #printmodedefault
51090   MOV R3,R1
51100   MOV R1,R0
51110   ADR R0,badparameters
51120   CMP R3,#0
51130   BEQ storeparameters
51140   SWI "OS_ReadUnsigned"
51150   BVS fail
51160   ADR R0,badmargin
51170   CMP R2,#79
51180   BGT fail
51190   MOV margin,R2
51200   CMP R3,#1
51210   BEQ storeparameters
51220   SWI "OS_ReadUnsigned"
51230   BVS fail
51240   ADR R0,badthreshold
51250   CMP R2,#255
51260   BGT fail
51270   MOV threshold,R2
51280   CMP R3,#2
51290   BEQ storeparameters
51300   SWI "OS_ReadUnsigned"
51310   BVS fail
51320   ADR R0,badprintmode
51330   CMP R2,#5
51340   BLT fail
51350   CMP R2,#7
51360   BGT fail
51370   CMP R2,#6
51380   BEQ fail
51390   MOV printmode,R2
51400   .storeparameters
51410   MOV R0,margin
51420   MOV R1,threshold
51430   MOV R2,printmode
51440   LDMFD (sp)!,{R3-R12,R14}
51450   ]
51460 NEXT pass%
51500 end%=P%
51540 =end%
```

File Search Utility

Jeff Shipp

This utility program is to be used for searching a disc from the currently selected directory down the directory structure for a specified file or directory. The filename may be wildcarded with either the '#' or '*' symbols. The text returned will be the pathname plus the filename of the file(s) or directory(ies) found.The program itself is written in BASIC and should be located in the $.Library directory so that it can be run when *Find <filename> is typed from any location on the disc. The program will not work properly if it is CHAINed because it reads the last *-command issued in order to extract the search file. Note also that it will over-write any existing program in memory.

When *Find <filename> is entered, the operating system first searches the current directory and then the $.Library directory (as defined in the Run$Path system variable: type *SHOW to see this) for the file "Find". Once it has located it, the filetype is read and compared with the system aliases. Since the program is type &FFB, i.e. BASIC, it substitutes the original command with *BASIC -quit "Find" <filename>. This comes from the system alias: Alias$@RunType_FFB. It means enter BASIC, run the program "Find" and then QUIT to the operating system. This command line also carries with it the search string, <filename>. This is extracted within the program by the SWI call OS_GetEnv which returns the address of where the *-command string is stored. It is then a relatively simple task to extract the search string. The program makes use of the SWI call OS_GBPB to read file information and search for a specified file.

The main procedure of the program (PROCread_dir) is recursive. After obtaining the search string, PROCread_dir is called, the variables path$ and dir$ represent the path to the current directory level and the name of the next directory down, respectively. Initially these are both empty as we are searching from our current level. The function, FNget_path, returns a string which is the pathname into the selected directory i.e. path$+"."+dir$. In line 240 another function, FNread_entries, is called. This has two parameters, the first one is set to 10 and selects the "read information" option of the SWI call, OS_GBPB, and the second one is the search string, which in this case is "*" so it will read information for all files. The variable returned, no_entries%, is the number of files and directories found in the directory, defined in $pathname%. If no_entries% is greater than zero then the directory is searched again only this time 9 is passed to FNread_entries which selects the read entries option, the search file being specified instead of the "*". If the number of files found returned is greater then zero then the pathname and each file found is printed out.

The program now searches for a directory in the current directory. If one is found, its name is extracted from buffer2% (written to by FNread_entries(10,"*")) and PROCread_dir is called recursively with the new pathname. Any search files found are printed out at this new directory level.

The procedure, PROCread_dir, will continue to search down the directory tree structure until it finds a directory with no directories in it where-upon it will return from PROCread_dir, go up one directory level and search the next directory. This search procedure will continue until the whole disc is searched from the current level.

A small quirk of the program is that on exit, it returns to supervisor mode. This is only a problem if it is run from BASIC and could be overcome by redefining the RunType_FFB system alias to `BASIC |"%0|" %*1`. But then if it is run from the supervisor mode, it will return to BASIC upon exit.

```
 10 REM > $.Library.Find
 20 REM File search utility
 30 REM Jeff Shipp, January 1988
 40 :
 50 DIM pathname% 256,filename% 12
                          ,name% 12
 60 DIM buffer1% 850  :REM Buffer
             to store files found
 70 DIM buffer2% 3080 :REM Buffer
          to store file attributes
 80 ON ERROR REPORT:PRINT:END
 90 :
100 SYS "OS_GetEnv" TO r0
110 offset=12
120 REPEAT
130   offset+=1
140 UNTIL r0?offset=&20
150 file$=FNname(r0+offset+1)
160 REM Enter main loop
170 PROCread_dir("","")
180 END
190 :
200 REM Main procedure
210 DEF PROCread_dir(path$,dir$)
220 LOCAL index%,posn%
230 $pathname%=FNget_path(path$,
                             dir$)
240 no_entries%=FNread_entries(10
                            ,"*")
250 IF no_entries%>0 THEN
260   REM If selected directory
          has anything in it....
270   no_found=FNread_entries(9,
                           file$)
280   file_count=0:offset=buffer1%
290   WHILE file_count<>no_found
300     REM Output file found to
          VDU if its name matches
310     IF $pathname%<>"" PRINT
                    $pathname%+".";
320     SYS "OS_Write0",offset TO
    offset
330     SYS "OS_NewLine"
340     file_count+=1
350   ENDWHILE
360   index%=0:posn%=&10
370   WHILE index%<>no_entries%
380     IF buffer2%?posn%=2 THEN
390       REM Get directory name
400       $name%=FNname(buffer2%+
                          posn%+4)
410       REM Look into directory
420       PROCread_dir(FNget_path
             (path$,dir$),$name%)
430       $pathname%=FNget_path
                     (path$,dir$)
440       REM Read entries for
                    previous level
450       no_entries%=
           FNread_entries(10,"*")
460     ENDIF
470     REM Calculate position of
                 next Object type
480     X%=posn%+4
490     REPEAT
500       X%+=1
510     UNTIL buffer2%?X%=0
520     posn%=&10+X%+(4-(X%MOD4))
530     index%+=1
540   ENDWHILE
550 ENDIF
560 ENDPROC
570 :
580 DEF FNget_path(A$,B$)
590 IF A$<>"" THEN = A$+"."+B$
                        ELSE = B$
600 :
610 DEF FNread_entries(reason,
                      $filename%)
620 IF reason=9 THEN
630   start=buffer1%
640   length=850
650   ELSE
660   start=buffer2%
670   length=3080
680 ENDIF
690 SYS"OS_GBPB",reason,pathname%
    ,start,77,0,length,filename%
                         TO ,,,r0
700 = r0
```

```
720 REM Gets filename or
                       directory name
730 DEFFNname(point%)
740 A$="":Y%=0
750 WHILE point%?Y%<>0
760   A$=A$+CHR$(point%?Y%)
770   Y%+=1
780 ENDWHILE
790 = A$
```

Submitting Articles to Archive

Various folk have asked about submitting articles to Archive. Here are a few pointers

1. Before submitting an article, it would be sensible to ring us and find out whether we want it. We may already have articles in the pipeline on your particular subject.

2. When submitting an article or even a longish hint, it would make our life much easier if it could be put on disc as well as on paper – preferably **not** on 800k format but any other format would do 5.25"/3.5" DFS/ADFS 40/80. (We've just bought one of those dual 5.25"/3.5" drives from Watford which we've put on the Master but we don't have Kermit on the Archimedes yet.) Put your name and address on the disc and there's a good chance that we'll send it back to you!

3. No real comments about style – we don't mind as long as it's readable. Just keep it short – "Say what you've got to say, then shut up" is my motto! I'm not the kind of person who rambles on and on and on without really saying anything of any consequence, as I'm sure you'll agree if you have read Archive editorial and other articles I have written in the past... Ooops!

Fact or Fantasy?

One of our readers, using information from various sources has come up with the following mixture of fact and speculation.

The 300 series is fully upgradable to the 400 specification! – all except for the hi-resolution monochrome modes, since they require extra hardware. The 1.2 OS allows for 4 podules, 4 Mbytes of ram, hard disc (using one of the podules) and a co-processor podule. This assumes you have got the 2 micron CPU not the older 3μ version which didn't have the co-processor logic or integer multiplier. The chip set, manufactured by VLSI, is available from Quarndon.

To get 4 Mbytes, you replace the 32-256k drams with 32-1Mb drams. To get 4 podules where one is a full spec podule to take a floating point co-processor and 3 are normal, just plug in a 400 series backplane! *(I thought the connector on the 300 series only had a 64 connections, not the full 96!? Mine has the middle rowof pins missing. Ed.)* The FPU increases some benchmarks by at least 10! Later CPUs may have the FPU on the chip – the ARM CPU can handle up to 16 co-processors!

Computers with 8 MHz ROMs will obviously work faster than the present 4 MHz ones. *(If you can afford to buy the chips! Ed.)* The IBM PC podule uses an 80186 CPU which has about twice the performance of a standard PC. *(But development is "on hold". Ed. Sorry to keep interrupting, by the way!)* The hard disc podule controls standard units, so again, with the right software, larger units could be fitted.

Acorn or Olivetti will be producing workstations using the Archimedes chips. They may use the CPU at full speed which gives about 16 to 18 MIPS! The 400 series will probably end up with a Unix operating system called ARX. This is a friendly version of UNIX where the machine interfaces to the user rather than the other way round. With luck, a 300 series with 4 Mbytes could also run UNIX thus keeping it supported in the future. *(It would be nice if all this were true. Ed.)*

Order Form

	Quantity	Total
Program Disc Nº1☐ Nº2☐ Nº3☐ Nº4☐ Nº5☐ (£3 each)		
Shareware Graphics Demonstration Disc £3		
Wabash 3.5" Discs Bulk pack – 10 for £16		
In a library case – 10 for £18		
"ARM Assembly Language Programming" £12		
Minerva Deltabase £26		
Minerva System Delta-Plus £64		
System Delta Plus Reference Guide £27		
Minerva Minotaur £13		
Clares' Archimedes Toolkit Module £37		
Clares' Graphic Writer (word-processor) £27		
Clares' Artisan Art Package £37		
Clares' Alpha-Base £46		
Archimedes Programmer's Reference Manual £28		
Acorn Backplane AKA 01 £42		
Acorn I/O Podule AKA 10 £84		
Acorn ROM Podule AKA 05 £63		
Computer Concepts ROM Podule £53 (with battery backup £63)		
Computer Concepts Inter-Sheet ROM £42		
Computer Concepts Inter-Chart ROM £27		
Computer Concepts Inter-Word ROM £42		
I enclose a cheque payable to "Norwich Computer Services" for: (Sorry no Access or Visa facilities.)		

All prices are inclusive of VAT, where applicable, and UK carriage.

Name ______________________________

Address ______________________________

____

____

*HELP Archive

What are its aims? – Archive is a subscription magazine for users of the Acorn Archimedes range of computers, which aims (1) to provide information to the user (2) to provide a forum in which we can all share ideas (3) to give the benefit of bulk buying of software (4) to allow software and hardware vendors to advertise their wares.

Is it a User Group? – No, but I would like it to have a "User Group feel" – like BEEBUG when it first started – except that Norwich Computer Services has to earn a living from the magazine – hence the order form overleaf.

However, Sue and I are not in this business to make lots of money; we enjoy the work we do and it's very satisfying to be able to provide a useful service. We only ask to make enough money to live on plus a bit to give away and then we'll be happy. (I hope it doesn't sound too trite, because it's true.)

HELP! – Could you help us, please, to make Archive a success? We don't have the muscle or the financial budget of the big magazines and cannot afford to do a vast amount of advertising, so, if you think Archive is good, please take out a full subscription (if you have not already done so) and recommend the magazine to a friend or two. If you want any more information sheets and subscription forms, please let us know.

Can we answer technical enquiries by phone?

As much as we would like to continue the policy we have always had of being available to answer all your technical enquiries by phone, we felt that if we were not careful, we would be so inundated with calls that we would not have time to get the information out to you through the magazine. For that reason, we have introduced the Technical Help Service so that those who really need the instant access to help can purchase it. (£8 per year.) I hope you will bear with us and, if you do not feel it is worth the extra £8, please send your enquiries by post.

Do we take Access or Visa? No, I'm afraid not – mainly because of the high percentage that we would have to pay for the privilege (I think it's 6% when you first start). The other reason is that we have always had a policy of sending goods out by return of post, whenever possible, regardless of whether the cheques have cleared through the bank. The way we see it is that if you trust us by sending a cheque without the Access guarantee, it is reasonable for us to trust you by sending out the goods without waiting for the cheque to clear. Perhaps we are foolish to take the risk, but we have only had two dud cheques in three years of full time trading.

Those who help us... Although there's only two of us here, we are surrounded by a lot of people who help us in various ways, some voluntarily and some professionally, without whom we would not be able to stay in business. I don't think it is appropriate to mention them all by name, but I would just like to assure them that we are really grateful to them all. However, as I have said in most of my previous publications, as a committed Christian, there is One Person who never fails us and without whom we would achieve nothing worthwhile. God supplies all our needs, and we thank and praise Him!

I hope you find Archive interesting and informative and that you will feel able to contribute your own ideas, hints and tips or even full articles. I'm afraid that we can't afford to pay vast sums for articles, but at least you'd automatically become an "HLMTHS". Honorary Life Member, Technical Help Service!

Thanks again,

Paul B.

Fact-File

CJE Micros	Dept AM, 78 Brighton Road, Worthing, W Sussex, BN11 2EN. (0903 – 213361)
Abacus Training	29 Okus Grove, Upper Stratton, Swindon, SN2 6QA.
Brainsoft	22 Baker Street, London, W1M 1DF. (01 – 486 – 0321)
Clares Micro Supplies	98 Middlewich Road, Rudheath, Northwich, Cheshire, CW9 7DA. (0606 – 48511)
Computer Concepts	Gaddesden Place, Hemel Hempstead, Herts, HP2 6EX. (0442 – 63933)
Datathorn Systems Ltd	50 Spring Grove, Loughton, Essex, IG10 4QD. (01 – 508 – 4904)
Contex Computing	15 Woodlands Close, Cople, Bedford, MK44 3UE. (02303 – 347)
EMR Ltd	14 Mount Close, Wickford, Essex, SS11 8HG. (0702 – 335747)
Fairhurst Instruments	Dean Court, Woodford Road, Wilmslow, SK9 2LT. (0625 – 525 – 694)
HS Software	56, Hendrefolian Avenue, Sketty, Swansea, SA2 7NB. (0792 – 204519)
IFEL	36 Upland Drive, Plymouth, Devon, PL6 6BD. (07555 – 7286)
Intelligent Interfaces	14 Julius Close, Chandlers Ford, Eastleigh, Hants, SO5 2AB. (04125 – 61514)
LTS Ltd	Haydon House, Alcester Road, Studley, Warks, B80 7AN. (0386 – 792617)
Meadow Computers	11, London Street, Whitchurch, Hants, RG28 7LH. (025689 – 2008)
Minerva Systems	69 Sidwell Street, Exeter, EX4 6PH. (0392 – 37756)
Northern Educational Software	16 Dawson Lane, Bierley, Bradford, BD4 6HN.
Pineapple Software	39 Brownlea Gardens, Seven Kings, Ilford, Essex, IG3 9NL. (01 – 599 – 1476)
RESOURCE	Exeter Road, Doncaster, DN2 4PY. (0302 – 63800/63784)
Serious Statistical Software	Lynwood, Benty Heath Lane, Willaston, South Wirral, L64 1SD. (051 – 327 – 4268)
Tubelink	P.O.Box 641, London, NW9 8TF.
Norwich Computer Services	**18 Mile End Road, Norwich, NR4 7QY. (0603 – 507057)**

Archive

The Subscription Magazine for *Archimedes* users

Articles include:

- File transfer on RS423
- Attaching a 5.25" drive
- Clares' Toolkit module
- Graphics demo programs
- Using screen banks
- Archie the Music Computer
- Structuring with BASIC V
- Technical notes on BASIC V
- Help with using the ADFS
- Epson screen dump
- Using BBC ROM images
- BBC micro as an I/O podule
- Assembly language programming
- Writing relocatable modules
- Using the WIMP environment
- Reviews of software and hardware

Technical Help Service (£8 / year)
A telephone hot-line service for immediate help with your technical problems. Any member can send written enquiries, but for a fast response use the THS!

Members' Discount: 7.5% off software from Computer Concepts, Minerva Systems and Clares Micros Supplies purchased through Norwich Computer Services.

Subscription: 12 issues £10 (UK/Eire) Europe £16, Middle East £20, America / Africa £23, Elsewhere £25. Technical Help Service £8

* Please send copies of *Archive* magazine for one year starting from
Vol. 1, Nº1 / Vol. 1, Nº2 / Vol. 1, Nº3 / Vol. 1, Nº4 / Vol. 1, Nº5 / Vol. 1, Nº6

* Please enrol me on the Technical Help Service for one year. (£8)

I enclose a cheque for £ ________

Name: ____________________

Address: ____________________

____

____________________ Postcode: ________

Norwich Computer Services, 18 Mile End Road, Norwich, NR4 7QY.
Subscription – £10 per year, Technical Help Service – £8 per year